Communication-Aware Hardware-Assisted MPI Overlap Engine

Mohammadreza Bayatpour(✉), Jahanzeb Hashmi Maqbool, Sourav Chakraborty, Kaushik Kandadi Suresh, Seyedeh Mahdieh Ghazimirsaeed, Bharath Ramesh, Hari Subramoni, and Dhabaleswar K. Panda

Ohio State University, Columbus, OH 43210, USA
{bayatpour.1,hashmi.29,Chakraborty.52,kandadisuresh.1,ghazimirsaeed.3, ramesh.113,subramoni.1,panda.2}@osu.edu

Abstract. Overlap of computation and communication is critical for good application-level performance. Modern high-performance networks offer Hardware-assisted tag matching and rendezvous offload to enable communication progress without involving the host CPU. However, hardware based offload cannot be used in many situations due to various hardware limitations and performance issues. Furthermore, hardware-based designs cannot provide good overlap for common communication patterns involving unexpected messages or non-contiguous datatypes. In this paper, we address these limitations by designing a communication-aware overlap engine for MPI that uses novel hardware-assisted and software-based solutions to extract overlap for both expected and unexpected messages. The proposed design adapts to the application's communication requirements including message size, datatype, and relative timing of processes using heuristics and history-driven predictions. We evaluate the proposed designs against state-of-the-art MPI libraries and show up to 41% and 22% reduction in latency for collective operations and stencil-based application kernels on 1024 and 128 nodes, respectively, as well as 23% improvement in communication performance of the P3DFFT application.

1 Introduction

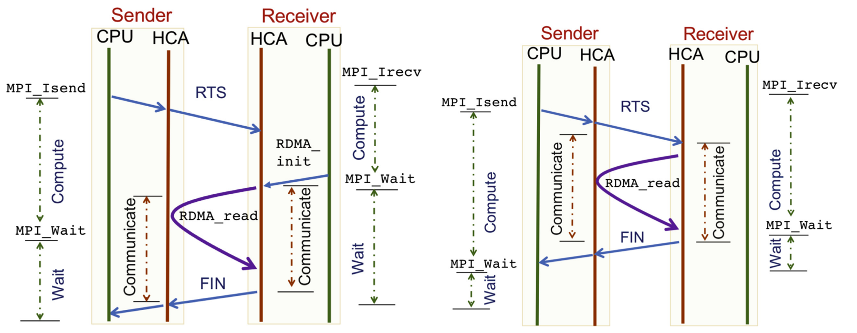
The massive growth in the size and scale of supercomputing systems has been driven by the current trends in multi-/many-core architectures and the availability of RDMA-enabled, and high-performance interconnects such as InfiniBand (IB) and Omni-Path. The Message Passing Interface (MPI) [18] has been the defacto programming model for developing high-performance parallel applications for the last couple of decades. One of the major features offered by modern high-performance network adapters (HCAs) is called 'RDMA' and it is the ability to

This research is supported in part by National Science Foundation grants #1931537, #1450440, #1664137, #1818253, and XRAC grant #NCR-130002.

P. Sadayappan et al. (Eds.): ISC High Performance 2020, LNCS 12151, pp. 517–535, 2020.
https://doi.org/10.1007/978-3-030-50743-5_26

read from and write data to remote memory locations without involving the host CPU. The MPI standard offers non-blocking communication primitives to take advantage of RDMA and enable overlap of communication and computation. Numerous studies have shown this overlap to be the critical factor for achieving good application performance and have proposed different solutions to address this [15,17,22].

There are generally two schemes in MPI to implement the point-to-point communications — 'Eager' and 'Rendezvous'. The eager protocol uses a set of pre-allocated and pre-registered buffers for the HCA to communicate asynchronously, without performing any handshake with peer processes. Upon receiving an eager message, this protocol involves one extra copy from pre-registered buffers into the application buffers, therefore, it is typically used for small messages. On the other hand, in Rendezvous protocol, the sending process first checks for the availability of the buffer in the receiver's side before transferring the actual message and it is used for medium and large message sizes. The Fig. 1(a) illustrates the RDMA read based rendezvous. The sender sends a control signal RTS to the receiver. The receiver after receiving the RTS issues an RDMA read signal which fetches the data from the sender without involving the sender's CPU. As it is seen in this figure, there is no overlap in communication with computation. In other words, the communication starts only after MPI wait is called by the application, after which the application is idle [25].



(a) Rendezvous RGET (b) Rendezvous RGET with HW TM

Fig. 1. Comparison of RGET with and without HW Tag Matching [5].

To tackle this, modern HCAs such as Mellanox Infiniband ConnectX-5 and ConnectX-6 have included the ability to perform tag-matching in hardware and initiate RDMA operations without the involvement of the CPU on the receiver side [5,26]. This allows the MPI library to post a receive operation along with the address of its destination buffer to the HCA. If the posted receive request is expected, meaning that the time that the receive request has been posted (t_{recv}) is before the time that incoming Tag Mathing (TM) packet has arrived (t_{arrive}), ($t_{arrive} > t_{recv}$), a matching receive for an incoming RTS gets offloaded to the

HCA. Therefore, the HCA can perform an *RDMA_read* from the sender's buffer as soon as it gets the RTS without any involvement of the receiver process. Rendezvous offload using Hardware Tag Matching is depicted in Fig. 1(b).

While this feature enables the MPI library to extract more overlap in certain scenarios, it cannot be used as a universal solution due to various semantic limitations and performance bottlenecks [5]. For instance, when no matching receive is found for an incoming message (unexpected message), it cannot be handled by the HCA since it does not know the destination buffer. Similarly, small messages or non-contiguous messages may not be offloaded to the HCA due to performance reasons. Even for expected messages, existing hardware-assisted solutions [5,26] do not provide the HCA peak bandwidth while the posted receive requests are offloaded. Furthermore, due to the semantic limitations of Hardware Tag Matching, these solutions do not provide reasonable overlap of communication and computation when the application uses a combination of 1) short and large or 2) contiguous and non-contiguous messages. These observations show that while hardware tag matching is useful in certain scenarios, MPI libraries need to address several challenges to mitigate its limited applicability as well as performance bottlenecks to provide a complete and high-performance solution.

2 Challenges

In this paper, our goal is to **design an overlap engine capable of adaptively utilizing advanced hardware and software-based schemes for progressing MPI operations for diverse application communication scenarios**. To achieve this, we need to answer the following five questions: **1)** What are the performance characteristics, benefits, and shortcomings of state-of-the-art hardware tag-matching and offload? **2)** Are the capabilities provided by the hardware sufficient or do they need to be augmented by software-based schemes? **3)** Which communication scenarios can be improved in terms of performance and overlap

Table 1. State-of-the-art designs and features to support efficient communication and computation overlap. In this table, we define the following design challenges for a high-performance overlap engine: C1) Adaptability to application communication requirements, C2) Efficient designs to extract overlap for unexpected messages, C3) Communication progress without receiver involvement, and C4) Efficient overlap for out-of-order messages

Design challenges	State-of-the-art MPI libraries				
	OpenMPI+ UCX with HW-TM	OpenMPI+ UCX	MVAPICH2	MVAPICH2+ Async	Proposed
C1 [see the caption]	✗	✗	✗	✗	✔
C2	✗	✗	✗	✗	✔
C3	✔	✗	✗	✔	✔
C4	✗	✗	✗	✗	✔

by offloading to HW? **4)** Can we propose novel designs to extract overlap for unexpected messages? Can it be done without increasing the memory footprint of the MPI library? **5)** Can the proposed designs be combined so they can be adaptively applied to the application's communication requirements?

Table 1 shows an overview of the state-of-the-art solutions available in different MPI libraries to extract overlap. Four different representative open source solutions are considered - OpenMPI+UCX with and without support for hardware tag-matching [26]; and MVAPICH2 with and without software-based asynchronous progress [21]. As we can see, both hardware and software-based solutions enable communication progress without receiver involvement for expected messages. However, none of the solutions provide good overlap for unexpected messages. Furthermore, even for expected messages, existing hardware-assisted solutions do not provide good overlap when the application uses a combination of 1) short and large or 2) contiguous and non-contiguous messages. Similarly, existing designs do not efficiently handle out-of-order messages. These issues limit the performance and applicability of the existing solutions. To the best of our knowledge, a comprehensive solution that adaptively and efficiently handles different application scenarios has neither been proposed in literature nor available as a software product.

3 Contributions

In this paper, we tackle these questions and show that neither hardware nor software-based tag matching can provide the best performance and overlap for different communication scenarios. Thus, a hybrid design that can take advantage of both these approaches and adapt to the application's communication requirements is required. To this end, we propose a Communication-Aware Hardware-Assisted MPI Overlap Engine ("CHAMPION") that takes advantage of hardware and software features to provide high overlap of computation and communication for both expected and unexpected messages, and dynamically adapt to the application's communication requirements. To summarize, the paper makes the following contributions:

- In-depth characterization of state-of-the-art hardware tag-matching and offload schemes and identify regions of applicability for hardware tag matching and software-based solutions.
- Design a communication-aware Hardware Tag Matching offload mechanism that hides the performance overheads of the offload engine while maintaining the peak performance of this engine.
- Enable the processing of out-of-order messages in hardware, using a trace-based matching design to maximize the benefits of Hardware Tag Matching.
- Propose novel designs to extract overlap from unexpected rendezvous messages by efficient prefetching.
- Evaluate and analyze the proposed designs using various benchmarks and application kernels.

Compared to state-of-the-art MPI libraries, the proposed designs show up to 41% improvement for collectives on 1024 nodes, up to 23% reduction in latency, and up to 2× improvement in overlap for a stencil-based application kernel on 128 nodes and 23% improvement in communication performance of P3DFFT application.

4 Motivation

As the first step toward designing a high-performance and scalable overlap engine inside the MPI library using Hardware Tag Matching, we need to systematically analyze the key communication primitives and semantics in the MPI library. We consider the semantic challenges as well as performance challenges of state-of-the-art Hardware Tag Matching. To have a complete picture of the hardware improvements in this technology, we analyze Hardware Tag Matching in two latest models of Infiniband HCAs: ConnectX-5 EDR and ConnectX-6 HDR.

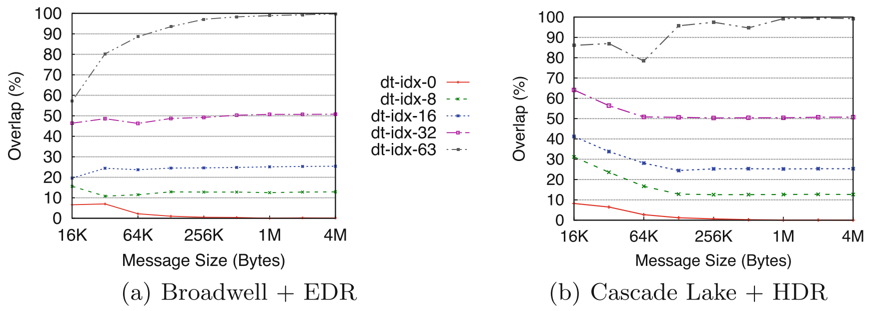


Fig. 2. Comparison of communication and computation overlap of Hardware Tag Matching with respect to the index of the inserted non-contiguous message in the window size of 64 on different architectures. This figure indicates that the receive requests which were posted after the non-contiguous receive request in the window are not getting offloaded, leading to underutilization of Hardware Tag Matching and lower overlap.

4.1 Semantic Challenges

In this section, to realize the impact of non-contiguous datatype on the overlap for medium to large messages, we modified the OMB [1] suite to insert a non-contiguous message in the window size of 64 and calculate the overlap and total latency. Figure 2 shows the impact of this insertion on the overlap. For instance,‘dt-idx-0’ shows that insertion of non-contiguous message as the first transfer in the window leads to total overlap of almost 0%. On the other hand,‘dt-idx-63’ which is the last transfer in the window almost has no impact on the expected overlap. These results indicate that posted receive messages that have been posted after the inserted non-contiguous message were not offloaded to the Hardware Tag Matching engine, leading to poor overlap and underutilization of Hardware Tag Matching. Such poor performance exists for both EDR as well as HDR models. MPI semantics mandate the MPI library to preserve the message ordering, i.e., consecutive messages with the same tag must match the posted receives in the same order they were posted. However, some of the posted receives cannot be offloaded to the HCA (such as unexpected messages or expected messages with non-contiguous datatypes) and need to be matched in software. In this scenario, the offloaded tag-matching handled by the HCA and software-based tag-matching handled by the CPU based communication progress in MPI library could lead to incorrect ordering. For instance, consider a scenario where the receiver posts two receives r_1, r_2, with the same tag t. However, due to some limitation r_1 cannot be offloaded to the HCA so only r_2 is offloaded. Now, the sender performs two sends s_1 and s_2 that should match with r_1 and r_2 respectively. However, the HCA will process the incoming message from s_1 first and match it to r_2, violating the MPI ordering semantics. Conversely, if the MPI library does not offload r_2 to prevent this scenario, it has to be progressed in software and benefits of hardware tag matching can not be obtained. **Thus, a high-performance MPI library should have the necessary designs to maximize the applicability of Hardware Tag Matching.**

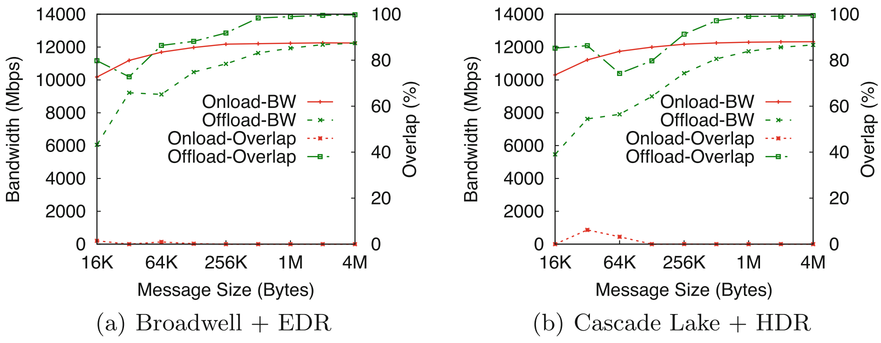


Fig. 3. Comparison of bandwidth and overlap for different message sizes with and without hardware rendezvous offload on different architectures. As we observe here, Hardware Tag Matching is able to provide higher overlap compared to the software-based solutions for point-to-point communications. On the other hand, this feature is unable to maintain the peak bandwidth for message range less than 1 MB. HDR and EDR are showing similar behavior.

4.2 Performance Challenges

In this section, we compare the communication performance (bandwidth) and overlap achieved using point-to-point operations in Fig. 3. To measure overlap, we modified the `osu_bw` benchmark to calculate overlap similar to nonblocking collectives in OMB [1]. We call this benchmark `osu_bw_overlap`. To observe the effect of expected messages we introduce artificial delays at the sender to make sure that all the messages are expected and are handled by Hardware Tag Matching. As shown in Fig. 3, Hardware Tag Matching maintains the peak overlap of communication and computation in both EDR and HDR architectures. However, in the 'onload' scenario, there is no overlap of communication and computation. Here we assume that there is no asynchronous process/thread performing the communication progress on behalf of the main process. Therefore, RDMA Read is initiated only after application calls MPI_Wait, which happens after computation is done. On the other hand, by comparing the bandwidth of the 'offload' vs. 'onload', we can realize that Hardware Tag Matching performs worse compared to the CPU onloading. In the case of unexpected messages, RDMA Read is always initiated by CPU, therefore, the bandwidth is same for both 'onload' and 'offload' scenarios. Hardware tag matching requires pre-posting receives to HCA before the message has arrived at the receiver, so that the HCA can directly put the data into the application buffer. Clearly, this scheme cannot be applied to unexpected messages, where the message has already arrived but no matching receive operation has been posted to the HCA. As illustrated in Fig. 7(a), this scenario prevents overlap of computation and communication at the receiver. Also, the delayed receiver process may increase the communication progress and wait time at the sender side, leading to the propagation of a skew from the receiver process to the sender process. Furthermore, since the HCA is unable to process unexpected messages independent of the CPU, it disables hardware tag matching once an unexpected message arrives to avoid ordering issues. Further receives cannot be posted to the HCA until the unexpected message has been processed by the CPU. Since the process arrival pattern of HPC applications are often skewed [27], unexpected messages are a common scenario. **Thus, a high-performance Hardware Tag Matching assisted offload design in MPI must avoid performance degradation of hardware rendezvous offload while maintaining the peak overlap during the application runtime for both expected and unexpected messages.**

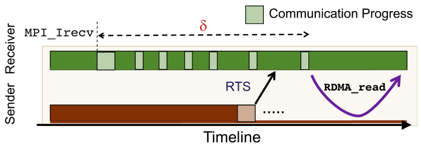
5 Proposed CHAMPION Design

A message in the MPI runtime can be classified as an expected or unexpected message. Each has its own challenges and requires different considerations to achieve better communication overlap. In view of the broad spectrum of MPI communication, we explore the design challenges and solutions for expected and unexpected messages. In the following sections, we show how our proposed designs for hardware tag-matching semantics augmented by software-based approaches are able to exploit better performance and overlap for various benchmarks and applications.

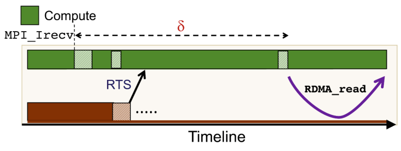
5.1 Communication-Aware and Adaptive Rendezvous HCA Offload

As we discussed in Sect. 4, rendezvous offload using Hardware Tag Matching has some performance degradations compared to the default version that all rendezvous protocol is initiated by host CPU. To address these limitations, we propose a communication aware design that tries to adaptively offload the MPI receive requests in an on-demand fashion. This design offloads only when an overlap opportunity is presented, otherwise, it avoids offloading to mitigate the overheads of the hardware tag-matching.

For expected messages, RDMA read needs to be performed as soon as the incoming RTS is received. To realize this in our opportunistic design, we employ a heuristic to find the frequency (f) of progress calls (`MPI_Wait`, `MPI_Test`) made during the runtime. The frequency f is computed based on the difference between the time (δ) when the receiver posts a receive request and when the actual RDMA operation is triggered. The frequency f has an inverse relation to the offloading factor e.g., the higher the progress frequency, the less the number of offloaded tag-matching requests. This adaptive progress design is opportunistic in nature as it continuously looks for overlap opportunities on the receiver side.



(a) High δ due to late sender



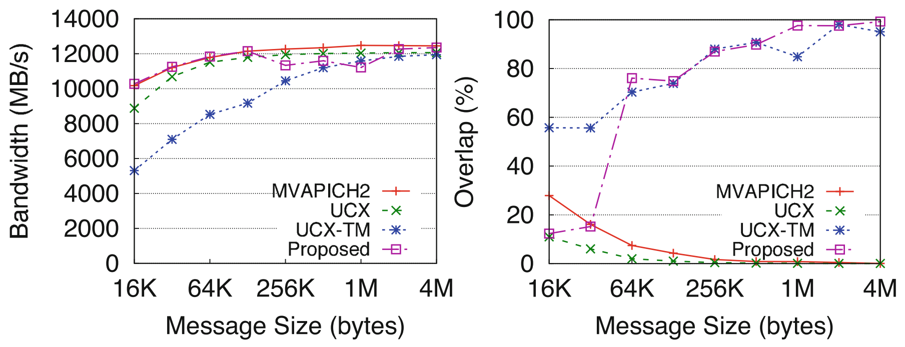
(b) High δ due to low frequency of communication progress

Fig. 4. In the proposed communication-aware Hardware Tag Matching design we consider high value of δ as indicator of the need for hardware offload. Here we show two different scenarios where δ can become high.

If δ is large enough (greater than a threshold K), then we try to offload the receive request for this process peer. K depends on the number of outstanding rendezvous requests (C_{rndv}) and the average network latency of all outstanding rendezvous messages. To approximate the latency, we use the LogGP [2] model. α and β are obtained in an offline fashion and they are architecture-specific.

$$K = \left(\frac{\sum_{i=1}^{C_{rndv}} (\alpha \times MSG_i + \beta)}{C_{rndv}}\right) \times C_{rndv} = \sum_{i=1}^{C_{rndv}} (\alpha \times MSG_i + \beta)$$

Large value of δ could be caused by either (or both) of the following cases: 1) Sender's RTS is posted later in time than the receiver has posted the receive-request. In this case, HW TM is needed to avoid the receiver to get blocked because of the late sender, leading to more overlap at the receiver side. Figure 4(a) depicts this scenario. 2) Receiver process does not progress the communication frequently enough and as a result, does not quickly poll the completion queue inside HCA to find the received RTS. In this scenario, HW TM is needed to take care of the handshake required for RGET based rendezvous protocol. This scenario is shown in Fig. 4(b). On the other hand, a small value of δ



(a) BW test using `osu_bw` (b) Overlap test using `osu_bw_overlap`

Fig. 5. Proposed communication-aware and adaptive HW-TM design on bandwidth and overlap benchmarks.

implies that there is no skew between the sender and the receiver processes. This means that the receiver process is frequently progressing the communication and there is no extra overlap gain from using HW TM, hence, we avoid offloading to the HCA. Our proposed communication-aware design further keeps track of the rate of canceled offloaded receive requests. If the cancelation rate passes a threshold during a period, our proposed design avoids offloading more receive requests until the next period. This is used to avoid the overheads of receive request cancelations.

Figures 5(a) and (b) show that our proposed design can correctly realize the lack of overlap opportunity in the `osu_bw` benchmark and adaptively avoids offloading receiver requests to the HCA. On the other hand, for `osu_bw_overlap` benchmark, it correctly offloads the receives to achieve maximum overlap. This benchmark is similar to `osu_bw` but it also calculates the overlap with the same formula as used in Nonblocking Collectives in OMB [1].

5.2 Trace-Based Matching

Scientific applications exhibit a wide variety of communication patterns involving a range of message sizes and datatype layouts. For instance, a sender is allowed to send a message with a derived datatype layout that cannot be offloaded, followed by a message with a contiguous datatype layout which can be offloaded. As discussed in Sect. 4, such variability in the message layouts limits the usability of the offloading if we only rely on HW TM semantics.

An MPI library must make sure that there are no messages offloaded to the network with the same tag to avoid the ordering issues which limit the usage of HW TM. To have such a capability, we add an additional variable to the tag-matching tuple of rank, tag, and context_id so that messages which have the same tag get differentiated. The new variable is unique and preserves the ordering of the messages for a sequence of messages with the same tag. To achieve this,

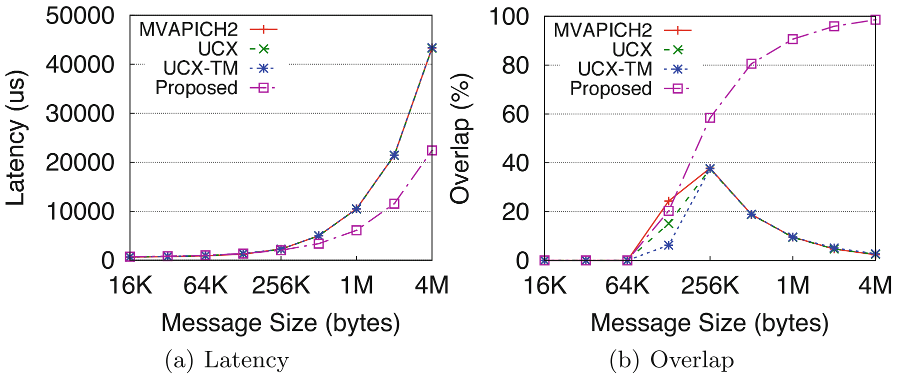


Fig. 6. Impact of proposed designs on variable memory layout communications. Out of 64 messages, all are using contiguous layouts while one is using MPI derived datatype.

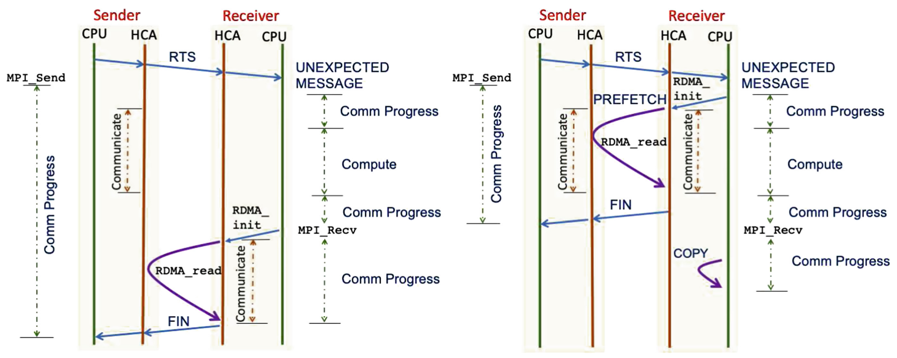
two sequence numbers are added for each communicating peer. If an application uses more than one unique 'tag' when it calls MPI *Send* and *Recv* primitives, we allocate new sequence numbers for each new tag. Every communicator in the application has its own set of sequence numbers so that context_id remains same for all the sequence numbers within a communicator. Both the sequence numbers are used to keep track of corresponding *Send* and *Recv* MPI calls issued by the application for each unique combination of rank, tag, and context_id. Whenever an application issues these operations, the sequence numbers for that specific tag and peer are incremented, as there is always a matching *Recv* operation for a *Send* operation. Appending this sequence number within the 64-bit value of HW TM tag ensures that no two messages can have the same tag, while MPI ordering of the messages is preserved.

To evaluate our design, we analyze the same benchmark that we used in Sect. 4.1. As we mentioned before, the benchmark creates an MPI derived datatype and during the `window_size` number of transfers, it runs few iterations with derived datatypes by using the same tag as of other transfers in each iteration. By running this benchmark, when a software-based pending receive is available, MPI libraries such as OpenMPI+UCX stop offloading new incoming receives to the HCA until software takes care of the pending receives. Due to this limitation, the hardware cannot be exploited to its full potential to achieve maximum overlap. Figure 6 shows how our proposed trace-based matching design overcomes this limitation. As it can be seen, the presence of even a single non-contiguous datatype transfer can completely eliminate the benefits of naive hardware tag-matching design (refer to Sect. 4). However, our proposed trace-based matching design is able to address these limitations and offer better performance and overlap in comparison to other state-of-the-art solutions.

5.3 Improving the Overlap of Unexpected Messages

In this section, we discuss various design components for our overlap engine which are applicable to '*unexpected messages*' at the receiver side. We start with a speculative approach and move towards an optimized design. As we discussed in Sect. 4, Hardware Tag Matching does not provide any communication and computation overlap for unexpected messages. This is an expected behavior as upon receiving an unexpected RTS, receiver process has not yet posted the receive request and therefore, receiver HCA does not yet know where the destination buffer is. As illustrated in Fig. 7(a), this scenario prevents overlap of computation and communication at the receiver. Also, the delayed receiver process may increase the communication progress and wait time at the sender side, leading to the propagation of a skew from the receiver process to the sender process. Since the process arrival pattern of HPC applications are often skewed [27], unexpected messages are a common scenario.

In order to allow the sender to proceed without getting stuck on a late receiver process to post the receive request and perform the RDMA-Read, the receiver process selectively prefetches some of the unexpected rendezvous messages. To achieve this, we create a memory pool and register it with HCA during `MPI_Init`. Whenever an unexpected RTS is received at the receive side, we query the memory pool to see if there is a memory slot available to be used for prefetching. If a free slot is found, then based on the sender's information obtained from RTS packet, an RDMA read is issued to transfer the data from the sender's buffer. After the completion of the transfer, a FIN packet is sent to the sender indicating that the sender is free to mark the operation as complete. This design is illustrated in Fig. 7(b). To decide whether or not prefetch the incoming unexpected RTS, this design relies on the history of the previous prefetches and number of useful prefetches (ones that receiver process posted the receive request after the



(a) Rendezvous RGET (b) Proposed Prefetch Scheme

Fig. 7. The proposed design and default MVAPICH2 approaches in handling unexpected rendezvous messages. In the proposed design, the receiver process prefetches the unexpected rendezvous message, leading to better overlap and latency for sender and receiver as the sender does not get blocked by the late receiver.

prefetch is done) versus non-useful prefetches (ones that receiver process posted the receive request before prefetch is done, but obviously, after RTS is received).

To measure the impact of the proposed design for unexpected messages, we use a synthetic benchmark where we inject skew between the sender and the receiver to force the message to arrive as unexpected at the receiver. As it is shown in Fig. 8, closest in performance is the MVAPICH2-X library with asynchronous progress thread enabled—referred to as MVAPICH2+Async. In our proposed design, we also create a `tm-thread` that functions similar to how MVAPICH2+Async functions [21]. The proposed design improved overall runtime by up to 38% as well as achieved better overlap in comparison to MVAPICH2+Async design.

6 Performance Evaluation

In this section, we discuss the experimental results of our proposed designs and provide in-depth analyses. We implemented our proposed designs in a publicly available open-source version of MVAPICH2 [19]. To evaluate the proposed designs, we provide an in-depth comparison against the state-of-the-art designs employed by MPI libraries such as MVAPICH2-X v2.3rc2 (referred to as "MVAPICH2") and Open MPI v4.0.0 with UCX v1.4 (referred to as "OpenMPI+UCX"). All the reported numbers are an average of five runs. Microbenchmark evaluations ran for 1,000 iterations for each run and an average of five runs is reported. Furthermore, the standard deviation between these iterations is kept under 5%.

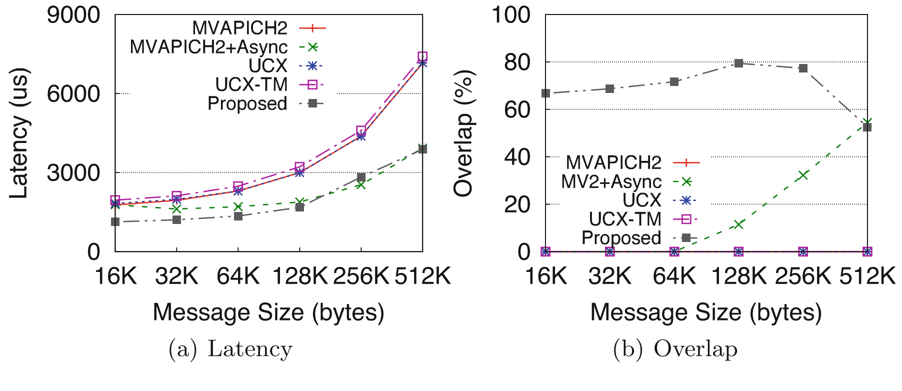


Fig. 8. Impact of the proposed prefetch-based design on the performance of unexpected messages (Window size = 64)

6.1 Experimental Setup

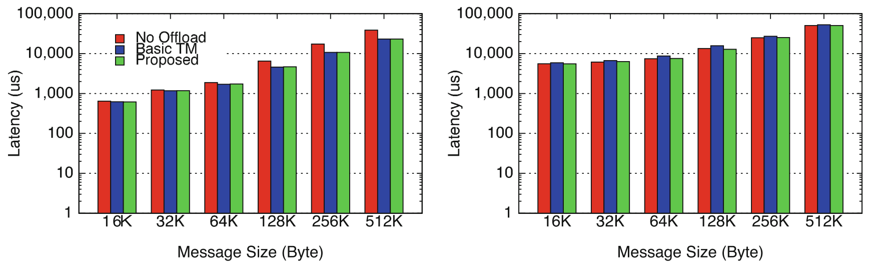
We used the following clusters for our evaluation:

Cluster-A—Frontera cluster at the Texas Advanced Computing Center contains 8008 compute nodes equipped with the dual-socket Intel Xeon Platinum 8280 (Cascade Lake), 56-core processors (448,448 cores in total) operating at 2.70 GHz with 192 GB RAM. Each node is equipped with Mellanox HDR-100 ConnectX-6 HCAs (100 Gbps data rate).

Cluster-B—Pitzer cluster at the Ohio Supercomputing Center contains 260 compute nodes equipped with the Skylake Gold 6148 series of Xeon dual-socket, 20-core processors operating at 2.40 GHz with 128 GB RAM. Each node is equipped with Mellanox MT4119 EDR ConnectX-5 HCAs with PCI-Ex Gen3 interfaces. For some of the motivational numbers we used our local cluster which has similar details to this cluster but it has Broadwell series of Xeon dual-socket, 14-core processors operating at 2.40 GHz. All the results were obtained on Cluster-B except for the ones which are indicated that they were run on Cluster-A.

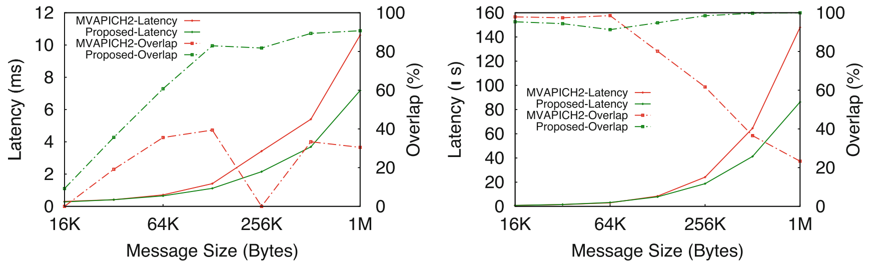
6.2 Impact of Proposed Designs on Collective Operations

In this section, we evaluate the performance of collective operations using the proposed designs.



(a) MPI_Iscatterv (b) MPI_Igatherv

Fig. 9. Impact of proposed communication-aware design on collectives with 640 processes.

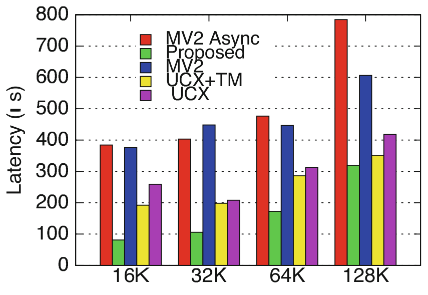


(a) MPI_Ialltoall (64 Nodes) (b) MPI_Iscatterv (1024 Nodes)

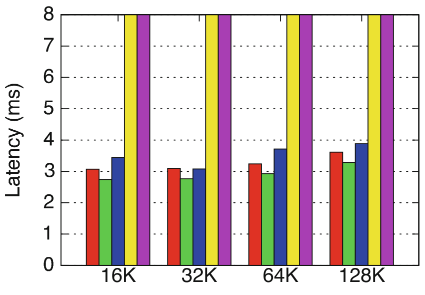
Fig. 10. Performance impact of proposed designs on MPI_Ialltoall and MPI_Iscatterv running on cluster A

Impact of Proposed Communication-Aware Hardware Tag Matching Design. Figure 9(a) shows that by using the proposed design, `MPI_Iscatterv` performance increases by a factor of 1.6X. In MVAPICH2, `MPI_Iscatterv` uses a direct algorithm, meaning that the root process directly sends the data to all other non-root processes. Therefore, HW TM can provide nearly perfect overlap of communication and computation for this collective as only one receive request is issued by non-roots during the collective runtime and this receive request gets overlapped using HW TM. On the other hand, Fig. 9(b) shows that for `MPI_Igatherv`, basic TM design has around 10% to 15% degradation compared to default for medium messages but the proposed design can avoid this degradation. After profiling this test, we realized that even though Igatherv uses a direct algorithm, more than 90% of the offloaded receive requests at root are getting canceled, therefore, there will be no benefit from HW TM. Since our communication-aware design keeps track of the cancel rate of the offloaded receive requests, it avoids using HW TM for this benchmark during the runtime leading higher performance compared to basic HW TM design.

Figure 10 shows the impact of the proposed designs at large scale for Ialltoall and Iscatterv collectives. As it is shown in these figures, for iscatterv in 1024 node, there is up to 41% improvement in total latency while increasing the overlap up to 80%. On the other hand, Ialltoall also shows up to 33% improvement in total latency and up to 70% in overlap providing near-perfect overlap of communication and computation on 64 nodes.



(a) MPI_Reduce

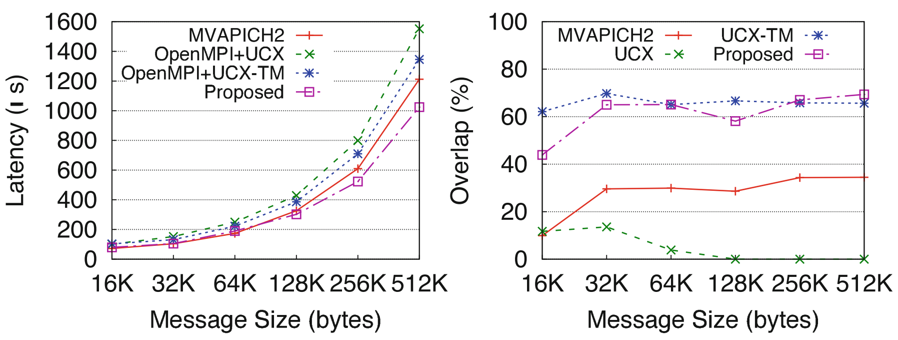


(b) MPI_Gather

Fig. 11. Impact of proposed prefetch-based design on collectives with 320 processes.

Impact of the Proposed Designs for Unexpected Messages. By running collective tests using OMB [1] using the prefetch-based design, we did not observe any significant difference in the performance of collective operations. This is expected because in collective benchmarks (OMB) all the processes are mostly synchronized e.g., all the processes enter the collective operation at the same time. To better understand the impact of our designs, we modify some benchmarks to insert skew for some of the processes as the real applications typically show skewed communication. We observed that the rooted collectives such as `MPI_Gather` and `MPI_Reduce` show benefits due to the prefetch design. This is due to the fact that the rooted collectives do not have any implicit barrier during the operation in contrast to dense collectives like `MPI_Alltoall`. For these communication patterns, the propagation of skew to other processes of the communicator can be avoided by efficiently prefetching the unexpected messages. Figure 11 conforms to our understanding where we see up to 55% improvement for `MPI_Reduce` and up to 17% improvement for `MPI_Gather` at 320 processes.

6.3 Impact of Proposed Designs on 3Dstencil Kernel

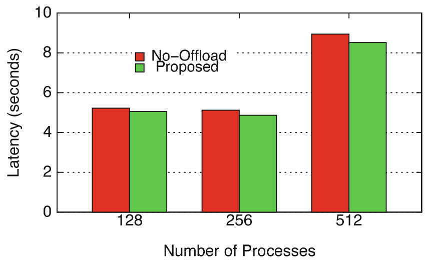


(a) Latency (b) Overlap

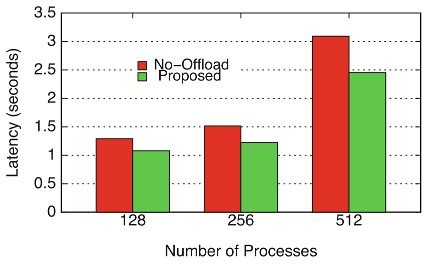
Fig. 12. Performance of 3D-Stencil application kernel running on 128 nodes using proposed designs on cluster A

3DStencil is a common communication kernel that mimics the communication pattern of many stencil-based applications and Adaptive Mesh Refinement (AMR) kernels. This communication kernel performs 7-point stencil with neighboring processes using MPI non-blocking point-to-point primitives i.e., `MPI_Isend` and `MPI_Irecv` using contiguous datatypes. Figure 12 shows that proposed design achieve up to 2× increased overlap and up to 22% improved latency as compared to other MPI libraries running on 128 nodes. In this experiment, for all the message sizes, the Rendezvous protocol is getting used and our profiles showed in the proposed design, out of 579,365 receive requests that have been offloaded, 20,050 receive request has been canceled while 559,315 offload requests have been successful, having the offload rate of 96%. This results in more than 20% improvement in overall runtime compared to default MVAPICH2.

6.4 Application-Level Evaluations



(a) Avg Total Time Per Forward-Backward FFT Loop



(b) Avg Communication Time Per Forward-Backward FFT Loop

Fig. 13. Impact of proposed designs on P3DFFT with 32 processes per node.

In this section, we evaluate the impact of the proposed designs on performance of P3DFFT and LAMMPS applications. The Parallel Three-Dimensional Fast Fourier Transforms (P3DFFT) [20] library uses a 2D, or pencil, decomposition and overcomes an important limitation to scalability inherent in FFT libraries by increasing the degree of parallelism. This library heavily relies on nonblocking Alltoall collectives to transform the data grid during each iteration [16,24]. Figure 13 shows the impact of the proposed design. As shown in this figure, the proposed design can correctly realize the opportunity of overlap in this application and provide 23% improvement in the communication time and up to 7% improvement in total application time.

Our second evaluation is on Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [3] which is a molecular dynamics program developed in Sandia National Laboratories. This test runs on 32 nodes with 16 processes per node on Frontera cluster. On this configuration, we observed 3.58% improvement in total execution time compared to MVAPICH2. We further profile this application and realized that out of 266,896 rendezvous recieves, 216,517 receive requests have been successfully offloaded and matched in the Hardware Tag Matching engine and rest have been handled by software tag matching. This leads to 81% success rate in the tag matching offload and improved overlap in the application.

7 Related Work

Optimizing software-based MPI tag matching has been the interest of many researchers. Some of these proposals [8,9,11] consider static designs to improve tag matching operations, while others [4,10,12,13] propose adaptive and dynamic approaches. Offloading the communication progress to NICs for MPI point-to-point and collective operations has been explored in the past. For example, Researchers [28] explore the implementation of multicast in Myrinet based NICs. Graham et al. [14] explore the overlap of computation and communication in Mellanox ConnectX2 HCA. It uses the Core-Direct API to implement the barrier collective and study the improvements in the total time obtained due to Hardware offloading. Subramoni et al. [23] provide designs to effectively implement the collectives on the ConnectX2 HCA. Brightwell et al. [6] showed that eagerly sending large messages can improve latency for pre-posted receives. However, this scheme has to resend unexpected large messages in the presence of application skew, which does not affect our design. Chakraborty et al. [7] investigate different approaches to increase the overlap of intra-node communication and computation with inter-node communication. As we can see, no work exists in literature that can provide maximum overlap of computation and communication in a communication-aware fashion while taking advantage of state-of-the-art solutions in hardware and software in an adaptive fashion as "CHAMPION" is able to do.

8 Conclusion and Future Work

In this paper, we characterized the semantic and performance limitations of state-of-the-art hardware-based tag matching and rendezvous offload designs and showed that they cannot be applied to a number of scenarios. We also show that hardware tag-matching does not provide improved overlap for various common communication patterns such as unexpected or non-contiguous messages. We proposed an adaptive overlap engine for MPI that is cognizant of the application's communication requirements and can opportunistically offload receives to the network adapter based on factors like message size, datatype, as well as arrival patterns of the sender and the receiver process. The proposed design uses both hardware-assisted and history-driven software-based solutions to extract overlap for both expected and unexpected messages in different communication scenarios. We evaluated the efficacy of the proposed design against state-of-the-art hardware and software-based solutions using a variety of microbenchmarks and application kernels and showed up to 55% and 17% improvement for Reduce and Gather collectives with 320 processes. Furthermore, we showed that our designs can increase the performance of Iscatterv and Ialltoall up to 41% and 33% in 1024 and 64 nodes, respectively. We also show up to $2\times$ increase in overlap and up to 22% reduction in overall runtime for stencil-based application kernels and 23% improvement in communication performance of P3DFFT. As the future work, we will work on proposing HW TM aware collectives as well as running more scientific applications to see the impact of proposed designs.

References

1. Osu Micro-benchmarks (2017). http://mvapich.cse.ohio-state.edu/benchmarks

2. Alexandrov, A., Ionescu, M.F., Schauser, K.E., Scheiman, C.: LogGP: incorporating long messages into the LogP model – one step closer towards a realistic model for parallel computation. Tech. rep., Santa Barbara, CA, USA (1995)
3. Atomic, L.S., Simulator, M.M.P.: LAMMPS (2013). https://lammps.sandia.gov/
4. Bayatpour, M., Subramoni, H., Chakraborty, S., Panda, D.K.: Adaptive and dynamic design for MPI tag matching. In: 2016 IEEE International Conference on Cluster Computing (CLUSTER), pp. 1–10 (September 2016). https://doi.org/10.1109/CLUSTER.2016.69
5. Bayatpour, M., Ghazimirsaeed, S.M., Xu, S., Subramoni, H., Panda, D.K.: Design and characterization of infiniband hardware tag matching in MPI. In: 20th Annual IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing (Accepted to be published) (2020)
6. Brightwell, R., Underwood, K.: Evaluation of an eager protocol optimization for MPI. In: Dongarra, J., Laforenza, D., Orlando, S. (eds.) EuroPVM/MPI 2003. LNCS, vol. 2840, pp. 327–334. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39924-7_46
7. Chakraborty, S., Bayatpour, M., Hashmi, J., Subramoni, H., Panda, D.K.: Cooperative rendezvous protocols for improved performance and overlap. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 361–373. IEEE (2018)
8. Dosanjh, M.G., et al.: The case for semi-permanent cache occupancy: understanding the impact of data locality on network processing. In: Proceedings of the 47th International Conference on Parallel Processing, p. 73. ACM (2018)
9. Dosanjh, M.G., Schonbein, W., Grant, R.E., Bridges, P.G., Ghazimirsaeed, S.M., Afsahi, A.: Fuzzy matching: hardware accelerated MPI communication middleware. In: 19th Annual IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing (CCGrid 2019) (2019)
10. Ghazimirsaeed, M., Grant, R., Afsahi, A.: A dynamic, unified design for dedicated message matching engines for collective and point-to-point communications. Parallel Comput. **89**, 102547 (2019)
11. Ghazimirsaeed, S.M., Afsahi, A.: Accelerating MPI message matching by a data clustering strategy. In: High Performance Computing Symposium (HPCS 2017). Kingston (2017)
12. Ghazimirsaeed, S.M., Grant, R.E., Afsahi, A.: A dedicated message matching mechanism for collective communications. In: Proceedings of the 47th International Conference on Parallel Processing Companion, p. 26. ACM (2018)
13. Ghazimirsaeed, S.M., Mirsadeghi, S.H., Afsahi, A.: Communication-aware message matching in MPI. Concurr. Comput.: Pract. Exp. **32**, e4862 (2019)
14. Graham, R.L., et al.: Overlapping computation and communication: barrier algorithms and ConnectX-2 Core-Direct capabilities. In: 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW), pp. 1–8 (April 2010). https://doi.org/10.1109/IPDPSW.2010.5470854
15. Hoefler, T., Siebert, C., Lumsdaine, A.: Group operation assembly language - a flexible way to express collective communication. In: ICPP-2009 - The 38th International Conference on Parallel Processing. IEEE (September 2009)
16. Kandalla, K., Subramoni, H., Tomko, K., Pekurovsky, D., Sur, S., Panda, D.K.: High-performance and scalable non-blocking all-to-all with collective offload on infiniband clusters: a study with parallel 3D FFT. Comput. Sci.-Res. Dev. **26**(3–4), 237 (2011)

17. Venkata, M., et al.: ConnectX-2 CORE-Direct enabled asynchronous broadcast collective communications. In: Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium, Workshops (2011)
18. Message Passing Interface Forum: MPI: A Message-Passing Interface Standard (March 1994)
19. MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE (2017). http://mvapich.cse.ohio-state.edu/
20. Pekurovsky, D.: P3DFFT: a framework for parallel computations of fourier transforms in three dimensions. SIAM J. Sci. Comput. **34**(4), C192–C209 (2012)
21. Ruhela, A., Subramoni, H., Chakraborty, S., Bayatpour, M., Kousha, P., Panda, D.K.D.: Efficient design for MPI asynchronous progress without dedicated resources. Parallel Comput. **85**, 13–26 (2019)
22. Schneider, T., Eckelmann, S., Hoefler, T., Rehm, W.: Kernel-based offload of collective operations – implementation, evaluation and lessons learned. In: Jeannot, E., Namyst, R., Roman, J. (eds.) Euro-Par 2011. LNCS, vol. 6853, pp. 264–275. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23397-5_26
23. Subramoni, H., Kandalla, K., Sur, S., Panda, D.K.: Design and evaluation of generalized collective communication primitives with overlap using ConnectX-2 offload engine. In: 2010 18th IEEE Symposium on High Performance Interconnects, pp. 40–49 (August 2010). https://doi.org/10.1109/HOTI.2010.22
24. Subramoni, H., et al.: Designing non-blocking personalized collectives with near perfect overlap for RDMA-enabled clusters. In: Kunkel, J.M., Ludwig, T. (eds.) ISC High Performance 2015. LNCS, vol. 9137, pp. 434–453. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20119-1_31
25. Subramoni, H., Chakraborty, S., Panda, D.K.: Designing dynamic and adaptive MPI point-to-point communication protocols for efficient overlap of computation and communication. In: Kunkel, J.M., Yokota, R., Balaji, P., Keyes, D. (eds.) ISC 2017. LNCS, vol. 10266, pp. 334–354. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58667-0_18
26. Unified Communication X (2019). http://www.openucx.org/
27. Venkatesh, A., et al.: A case for application-oblivious energy-efficient MPI runtime. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, pp. 1–12. IEEE (2015)
28. Yu, W., Buntinas, D., Panda, D.K.: High performance and reliable NIC-based multicast over Myrinet/GM-2. In: Proceedings of 2003 International Conference on Parallel Processing, 2003, pp. 197–204 (October 2003). https://doi.org/10.1109/ICPP.2003.1240581