Toward Developing a Unimem OFI Provider for MPI Support

Kyunghun Kim, Antonio J. Peña, Paul Carpenter
Barcelona Supercomputing Center (BSC)
first.last@bsc.es

Polydoros Petrakis, Manolis Ploumidis, Nikolaos Dimou
Foundation for Research and Technology
(ppetrak,ploumid,maraz)@ics.forth.gr

Yanfei Guo, Kenneth Raffenetti, Pavan Balaji
Argonne National Laboratory
(yguo,raffenet,balaji)@anl.gov

ABSTRACT

OpenFabrics Interfaces (OFI) is a unified abstraction of fabric network hardware for high-performance computing (HPC). Recently, HPC runtimes such as MPICH have been written on top of OFI's libfabric API. In this work we present our ongoing efforts toward developing an OFI provider for the Unimem architecture to offer efficient MPI capabilities on this emerging technology. The MPICH CH4-based runtime running the current version of our OFI provider attains 14.47 µs latency for small messages and 360.59 MB/s peak bandwidth for large messages on a prototype Unimem board.

1 INTRODUCTION

Unimem is an ARM-based exascale architecture for high performance computing. As the name *Unimem* suggests, this architecture provides unified memory addressing across multiple nodes and it fits very well with the PGAS (Partitioned Global Address Space) model. However, there is certainly demand for MPI (Message Passing Interface) support on Unimem-based systems, due to the ever-lasting omnipotent MPI ecosystem. In this work we detail our efforts toward providing an efficient MPI runtime for the Unimem architecture.

MPI is a *de-facto* standard for high-performance cluster computing. During the more than 25 years of history of MPI, there have been a significant number of efforts on porting and optimizing MPI implementations to several high-performance networking technologies. In order to support multiple of these interconnects, major MPI implementations, such as MPICH or Open MPI, engineered a modular structure. The current trend from MPI runtimes to support multiple networking hardware, however, is to leverage third-party middleware for low-level networking abstraction, such as libfabric or UCX.

Libfabric is the core component of the OpenFabrics Interface (OFI) [2]. OFI is a framework to export fabric communication services, and libfabric provides a unified user-space application programming interface (API), which is typically used by frameworks such as MPI or SHMEM, whereas each hardware vendor implements its OFI provider to support its specific functionality.

Following the state–of–the–art practice, our solution to provide an efficient and fully-functional MPI runtime for the Unimem architecture is developing an OFI provider, which is initially targeting the under-development MPICH CH4 initiative, due to our familiarity with this software project. We started implementing only the minimum set of OFI features mandatory to operate MPICH CH4, in order to subsequently apply performance optimizations. In this work we describe our initial design and implementation, along with the first optimizations we developed. We also provide an early performance evaluation.

EuroMPI'18, September 2018, Barcelona, Spain

2 BACKGROUND

2.1 Unimem APIs

Based on the Unimem architecture, Unimem user-space libraries offers two user-level APIs: virtualized mailboxes for low latency atomic message delivery and RDMA capabilities. The virtualized mailbox (vmbox) exposes a hardware queue for small data transfers, which is useful to deploy control messages. This mailbox transfers 192-bit fixed size messages to a given node in the network.

Unimem userspace library offers RDMA capabilities. Unimem DMA supports direct transfers from memory of one node to another node and guarantees high throughput. There are two main restrictions for Unimem DMA transfers due to hardware design: (1) since the Unimem architecture supports RDMA only for a specific memory address range, it is necessary to allocate buffers for transfers with a special function provided by the Unimem DMA API; and (2) before performing an RDMA read or write to a certain *remote virtual address*, the remote address should be registered to the local RDMA engine to prepare the route for the transfer.

2.2 libfabric for MPICH

libfabric is a generic framework aimed at covering functionality from a wide range of fabric hardware. It is designed in a modular fashion, where providers are free to implement the different existing modules. Before a libfabric user may leverage the functionality of a given module, it must query the implementation for specific support. MPICH checks for the OFI provider capabilities during the initialization process of the OFI netmod.

A fully-functional MPICH CH4 OFI netmod only requires the support of `FI_MSG`—basic message passing—and the `FI_MULTI_RECV` feature, which allows receiving multiple messages within a single large receive buffer. With these minimal libfabric features, we can operate MPICH entirely in *Active Message* mode, which provides a fully functional MPI implementation, with the tradeoff for overhead from the lack of native one-sided support, which we will target in future work.

3 DESIGN AND IMPLEMENTATION

Since Unimem DMA supports RMA read and write for *dmable* buffers only—those allocated through the `alloc_dmable_buf` Unimem allocator function—we started adding support for any type of user-provided buffer, including those allocated via any other

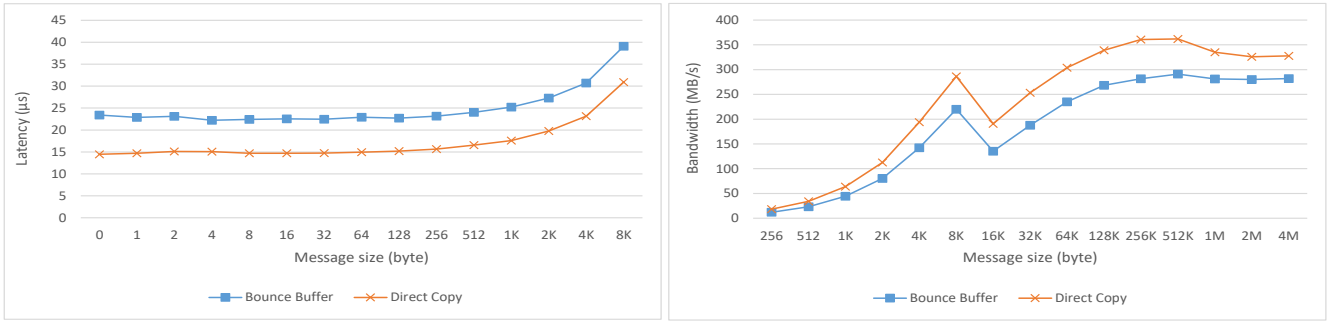

Figure 1: Left: Latency performance from `osu_latency` benchmark. Right: Bandwidth performance from `osu_bw` benchmark.

functions (non-dmable), focusing on functionality instead of performance. Hence, we implemented first the slow path using intermediate dmable bounce buffers at OFI provider level, leveraging the following operations: (1) copy from user buffer to sender-side bounce buffer, (2) RMA transfer from sender bounce buffer to receiver bounce buffer, (3) copy from receiver bounce buffer to MPI active message buffer, and (4) copy from receiver bounce buffer to user buffer.

We implemented the basic features of an OFI provider: *fabric*, *domain*, and *endpoints*. Our OFI provider supports connection-less endpoint type `FI_RDM`, required by MPICH, and manages connections for each node internally. We also implemented the basic necessary OFI constructs: *event queue*, *completion queue*, and *address vector*.

After attaining a fully-functional OFI provider under MPICH CH4, we started implementing optimizations. The mandatory *remote* virtual address registration described in Section 2.1 requires more than 10 µs. Since registering remote target buffers for each transfer poses an unbearable overhead for HPC use, we implemented a *registration cache*. In addition, we removed one of the two data copies at the receiver side from the naive implementation by allocating MPI active message buffers as dmable and exposing them to OFI, which significantly improved latency as shown in Section 4.

4 EVALUATION

We measured the performance of our current implementation with the OSU microbenchmarks [4]. Figure 1 shows the results of `osu_latency` and `osu_bw`. By preventing the extra copy on the receiver side described in Section 3 we were able to significantly improve latency, moving from 23.41 µs to 14.47 µs for 0-byte transfers while reaching a peak bandwidth of 360.59 MB/s.

However, this implementation still features a significant overhead compared to the performance of raw vmbox (< 1 µs) and DMA transfers (> 1 GB/s), mainly due to the absence of support for native RMA and direct transfers from dmable user buffers, which we leave for future work.

5 RELATED WORK

There have been many efforts on developing optimized MPI implementations for a specific networking architecture. To name a few, we can find a high-performance RDMA-based MPI implementation over InfiniBand, the precursor of MVAPICH [3], or a Portals 4 netmod for MPICH [5]. On the OFI approach, we may find an example in the implementation of a GNI OFI provider for the Cray XC dragonfly network and associated uGNI software stack [1].

6 CONCLUSIONS

We are developing an OFI provider for the Unimem architecture. We utilized virtualized mailbox for control messages and Unimem DMA to transfer message bodies to implement our message passing protocol. We applied our OFI provider to the MPICH CH4 runtime and evaluated its performance with OSU microbenchmarks. Our current implementation features 14.47 µs one-sided latency for zero-sized messages and 360.59 MB/s peak bandwidth for large messages.

For future work we need additional optimizations. For example, small enough messages could be transferred by directly writing to pre-allocated buffers in the receiver side (eager transfers), instead of ping-pong RTS/CTS control messages among nodes (rendezvous transfers). It is also possible to send smaller messages embedded within a single vmbox transfer. Supporting native RMA will further increase latency and bandwidth.

ACKNOWLEDGMENTS

This research is part of a project that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreements No. 754337, 749516, and 671578. This material includes support by the U.S. Dept. of Energy, Office of Science, Advanced Scientific Computing Research (SC-21), under contract DE-AC02-06CH11357.

REFERENCES

[1] Sung-Eun Choi, Howard Pritchard, James Shimek, James Swaro, Zachary Tiffany, and Ben Turrubiates. 2015. An implementation of OFI libfabric in support of multithreaded PGAS solutions. In *9th International Conference on Partitioned Global Address Space Programming Models (PGAS)*. IEEE, 59–69.

[2] Paul Grun, Sean Hefty, Sayantan Sur, David Goodell, Robert D. Russell, Howard Pritchard, and Jeffrey M. Squyres. 2015. A brief introduction to the OpenFabrics interfaces—A new network API for maximizing high performance application efficiency. In *23rd Annual Symposium on High-Performance Interconnects (HOTI)*.

[3] Jiuxing Liu, Jiesheng Wu, and Dhabaleswar K Panda. 2004. High performance RDMA-based MPI implementation over InfiniBand. *International Journal of Parallel Programming* 32, 3 (2004), 167–198.

[4] OSU Micro-Benchmarks. 2018. Osu network-based computing laboratory. http://mvapich.cse.ohio-state.edu/benchmarks. (2018).

[5] K. Raffenetti, A. J. Peña, and P. Balaji. 2015. Toward implementing robust support for Portals 4 networks in MPICH. In *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. 1173–1176.