2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) | 978-1-6654-9747-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/IPDPSW55747.2022.00083

Arm meets Cloud: A Case Study of MPI Library Performance on AWS Arm-based HPC Cloud with Elastic Fabric Adapter

Shulei Xu, Aamir Shafi, Hari Subramoni, and Dhabaleswar K. (DK) Panda

The Ohio State University
395 Dreese Laboratories, 2015 Neil Avenue
Columbus, Ohio
{xu.2452, shafi.16, subramoni.1, panda.2}@osu.edu

***Abstract*—Recent advances in HPC Cloud field has made multi-core high performance VM services more accessible. Emerging Arm based HPC systems are also receiving more attention. Amazon Web Service recently announced new c6gn instances with Gravition 2 Arm CPU on each node and support of Elastic Fabric Adapter, which make them the leading high performance Arm-based cloud system vendor. In this paper, we characterize the performance and capability of the AWS Arm architecture. We explore the performance optimization of current MPI libraries based on features of Arm-based cloud systems and Scalable Reliable Datagram protocol of Elastic Fabric Adapter and evaluate the impact of our optimization of high-performance MPI libraries. Our study shows that the performance optimization for MPI library on AWS Arm systems significantly improves the performance of MPI communication for both benchmark and application level. We gain up to 86% performance improvement in micro-benchmark level collective communication operations and up to 9% improvement in Weather Research and Forecasting application level. This paper provides a comprehensive performance evaluation for several popular MPI libraries on AWS Arm-based Cloud systems with EFA support. HPC application developers and users are able to get insights from our study to achieve better performance of their applications on Arm-based cloud systems with EFA support.**

***Index Terms*—HPC Cloud, MPI, Arm, Elastic Fabric Adapter, Scalable Reliable Datagram**

I. Introduction and Motivation

The recent development of High Performance Cloud technology has started an evolution to the Parallel and High Performance Computing fields. Vendors such as Amazon Web Services, Microsoft Azure, and Oracle Compute Infrastructure now provide remote virtualized compute resources with good support of high performance software and hardware stacks, which are significantly attractive to HPC application developers or users. As those vendors provide High Performance Cloud service with remote virtual compute system of different architectures, the interconnect components are normally built with either Infiniband RoCEv2 or their self-developed network adapter such as Elastic Fabric Adapter of AWS.

For most of the high performance cloud services in the market, vendors usually provide x86 architecture systems with Intel or AMD CPU. However, Amazon Web Services recently started to provide their Arm-based high performance cloud systems. These Arm-based cloud systems are built with their own-designed AWS Graviton processors, which has general support of high performance compute workloads.

Message Passing Interface [1] (MPI), as the de-facto programming model for large-scale parallel applications, is also commonly supported on different high performance cloud systems. MPI supports different communication patterns including point-to-point and collective communications.

With the emergence of rich set of cloud environments including different hardware or software stacks, MPI libraries should be optimized and tuned based on the features provided by cloud environments. However, there has no performance optimization work been done for MPI libraries on any state-of-the-art high performance Arm-based cloud systems.

In this paper, we first explore the performance optimization and enhancement of MPI libraries on the AWS Arm-based HPC cloud systems with EFA. Due to the nature of HPC cloud system, users have more accessibility and deeper control of the system. Our exploration of performance tuning and optional build setting with XPMEM kernel modules is capable of taking full advantage of features of Arm-based high performance cloud systems, in order to maximizing MPI application performance on Arm-based high performance cloud systems.

The high performance cloud service is becoming a more popular choice for HPC application developers and users. The emerging Arm processors have started their story in HPC field as well. Comparing to the current mainstream x86 systems, how would Arm-based cloud system perform differently? How can we bring HPC applications from x86 system to Arm-based system and keep the best efficiency? What performance and scalability should we expect for these HPC applications on Arm-based cloud systems? To answer those questions, this paper also provides a comprehensive performance characterization and evaluation of different MPI libraries on AWS Arm-based HPC Cloud system, including the latency and bandwidth comparing of different MPI communication operations, in both micro-benchmark level and application level.

978-1-6654-9747-3/22/$31.00 ©2022 IEEE
DOI 10.1109/IPDPSW55747.2022.00083

II. Background

A. Elastic Fabric Adapter

Amazon Elastic Compute Cloud (EC2) is a web service that provides resizable compute capacity in the cloud. An EC2 instance is a virtual server that can be used to run applications in AWS. The application developers can customize the instance features based on their budget and requirements. The on-demand allocation of resources is possible in cloud computing models due to their elastic capability and leads to cost-effective and energy-efficient solutions. Amazon Elastic Compute Cloud (EC2) provides a specialized Cluster Compute (CC) instance type to run HPC applications. The CC platform offers powerful and up-to-date CPUs and GPUs and an improved interconnection network well suited for large-scale HPC applications.

Elastic Fabric Adapter (EFA) is a network interface from Amazon for AWS EC2 instances. It is used to run applications that require high levels of inter-node communications at scale on AWS. It improves the performance of inter-node communications by avoiding the involvement of the Operating System (OS) kernel for each communication. EFA is an optional EC2 networking feature that can be enabled on any supported EC2 instance. For running MPI applications on AWS, the MPI library should support EFA for inter-node communications.

In the InfiniBand standard, there are four different transport modes: RC (Reliable Connected), RD (Reliable Datagram), UC (Unreliable Connected), and UD (Unreliable Datagram). These transport modes has different functionality and characteristics. Among these transport modes, EFA supports UD which is a connection-less and unreliable transport. More specifically, UD does not provide reliable delivery or ordering. Moreover, the maximum message size is limited to 1 Message Transfer Unit (MTU) which is 4KB. Thus, for messages larger than one MTU, the software should take care of reliability through re-transmissions, segmentation, and reassembly.

EFA supports a new transport mode other than UD called Scalable Reliable Datagram (SRD). SRD is similar to UD, but in different, it provides reliable delivery. It means that in SRD, the packets will be guaranteed to be delivered from sender to receiver, but since message are transferred through multiple paths, they might arrive out-of-order at the receiver.

B. Amazon Gravition 2

AWS Graviton 2 is the second generation of AWS own designed high performance Arm processor. It is a 64-core ARMv8 SoC custom-built by AWS using 64-bit Arm Neoverse cores. On AWS c6gn instances with Graviton 2 processor, the Elastic Fabric Adapter is supported with up to 100 Gbps interconnecting network bandwidth.

C. MPI

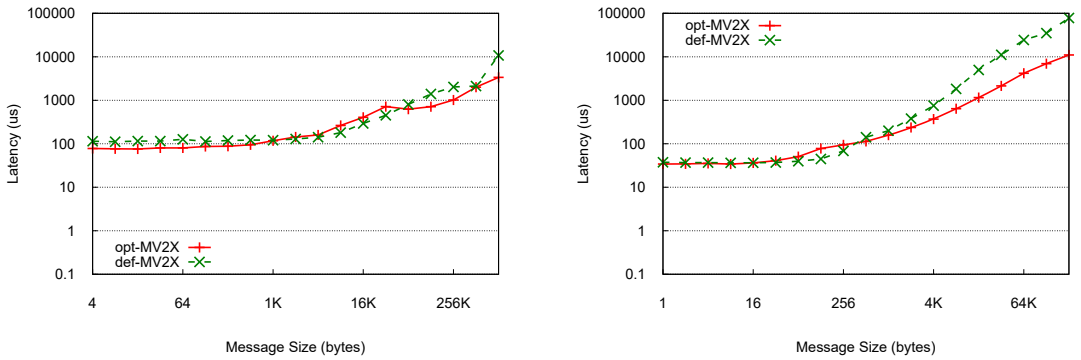
Message Passing Interface (MPI) is one of the most popular programming models for writing parallel applications in cluster computing area. MPI libraries provide basic communication support for a parallel computing job. In particular, several convenient point-to-point and collective communication operations are provided. High performance MPI implementations are closely tied to the underlying network dynamics and try to leverage the best communication performance on the given interconnect. MPICH [2], MVAPICH [3], Open MPI [4] and Intel MPI [5] are ones of popular MPI libraries.

III. The Optimization on AWS EC2 Cloud Arm Instances with EFA

Due to the popularity of MPI programming model among HPC users, most HPC cloud systems like AWS HPC clouds have started providing support for MPI libraries. In the parallelcluster of AWS EC2, some MPI libraries such as IntelMPI or OpenMPI libraries are supported as built-in modules of default instance image. Different to traditional supercomputer clusters, MPI libraries on the cloud systems are usually in different levels of optimization because of the highly customizable hardware or software configuration of cloud systems. For example, the MVAPICH2-X-AWS library share the same collective tuning settings for x86 Intel, AMD and Arm CPU systems. In AWS HPC clouds x86 instances, IntelMPI, OpenMPI and MVAPICH2-X are all supported. However in AWS Arm instances, there is no built-in support for IntelMPI. Therefore, we pick MVAPICH2-X-AWS as the MPI library to explore performance optimization and enhancement, and compare the optimized MVAPICH2-X-AWS performance with its unoptimized version, and later in the next section we have comprehensive performance evaluation across MVAPICH2-X and OpenMPI libraries.

The performance optimization and enhancement for MVAPICH2-X-AWS on AWS Arm-based HPC instances includes two main steps. The first step is performance tuning. Similarly to how HPC application users or developers always do, we adjust the mpirun parameters, and compare the performance of different collective algorithm selection through run-time arguments. After iterations of comparison, we could finalize the parameters and algorithms that achieve best performance on AWS Arm-based HPC instances, and this step will be iterated for each combination of number of nodes X processes per node. Second step is to take advantage of the higher freedom of loading kernel modules on Cloud systems. Compare to traditional supercomputer clusters, the HPC cloud service vendors allow users to have root access to the system. Some MPI libraries also have build option for these kernel modules. For example, both MVAPICH2-X and OpenMPI have optional build support with XPMEM. In order to have further optimization on AWS Arm-based instances, we load the XPMEM kernel module, and have MVAPICH2-X-AWS built with auto-detection of XPMEM module using dlopen.

Figure 1(a) and 1(b) shows an example of the benefit we gain from our performance optimization. On 16 nodes scale 64 ppn comparison, we observe 69% lower allreduce latency and 86% lower scatter latency with the mentioned tuning and optimization work.



(a) allreduce (b) scatter

Fig. 1: Example of Performance difference between optimized and unoptimized MVAPICH2-X-AWS on 16 c6gn instances scale with full subscription and EFA support

IV. Performance Evaluation

After the performance optimization for AWS HPC Arm system, we conduct a comprehensive performance evaluation on both mirco-benchmark level and application level. We compare the MVAPICH2-X-AWS with OpenMPI as the built-in MPI library of AWS parallelcluster [6]. Since in the previous section we have already done the optimization, we are using optimized MVAPICH2-X-AWS in this section.

A. Experimental Setup

Table I lists the hardware and software details of system configuration of the platform where we perform our experiments on.

TABLE I: Hardware & software specification of tested clusters

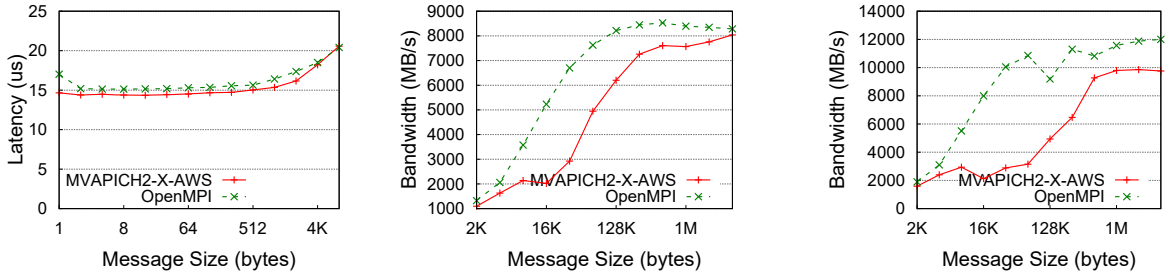
Specification	**AWS Arm**
Instance Type	c6gn.16xlarge
Processor	Amazon Graviton Gen 2
Clock Speed	2.5 GHz
#. Sockets	1
Cores Per socket	64
RAM (DDR4)	128 GB
Libfabric ver.	1.13.2
Parallelcluster	3.0.2
MVAPICH2	MVAPICH2-X-AWS
OpenMPI	4.1.0

B. Microbenchmark Results

We first do the performance evaluation with OSU-Microbenchmarks-5.8 [7].

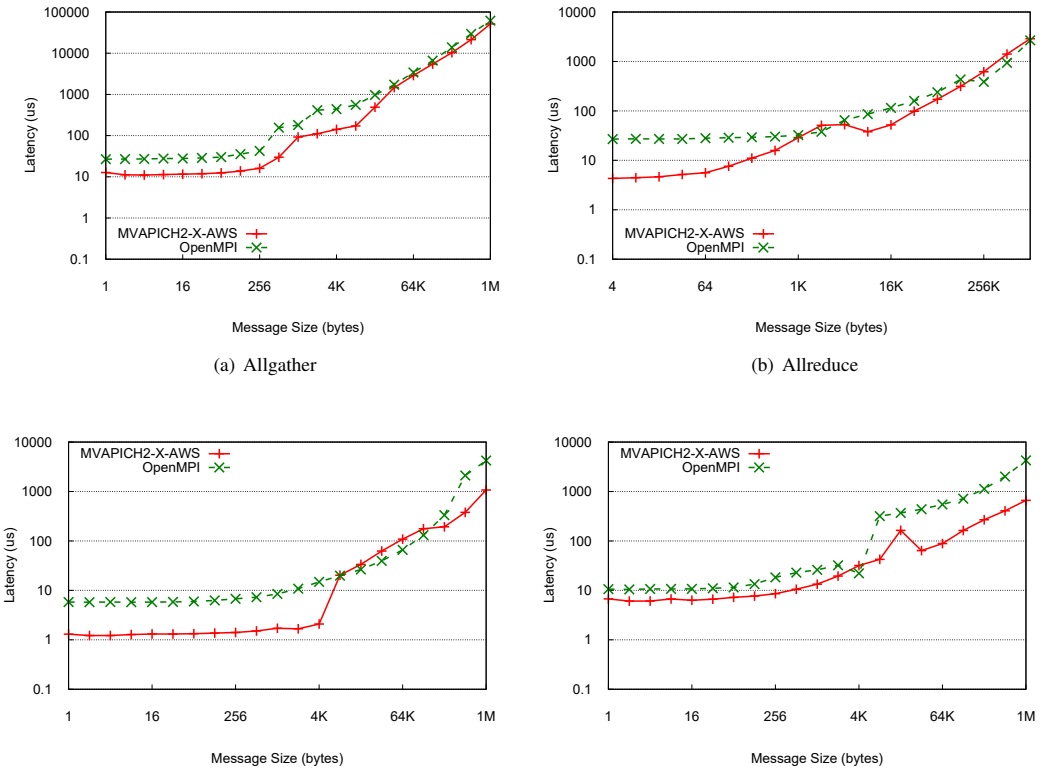
Figure 2 presents the point-to-point latency, bandwidth and bi-directional bandwidth of the inter-node MPI communication performance among two AWS Arm c6gn instances. We compare the inter-node latency, bandwidth and bi-directional bandwidth performance between optimized MVAPICH2-X-AWS and OpenMPI libraries. In figure 2(a), for small-size messages, MVAPICH2-X-AWS and OpenMPI has very close performance, while MVAPICH2-X-AWS has slightly lower latency, that is because OpenMPI are based on libfabric on AWS instances with EFA, while MVAPICH2-X-AWS directly ultilizes Scalable Reliable Datagram. In figure 2(b) and 2(c), we observe OpenMPI has higher bandwidth. That is mainly because the EFA verbs based SRD zero-copy design of MVAPICH2-X-AWS has more overheads with medium-sized messages, while OpenMPI is based on libfabric on AWS EFA.

After point-to-point latency and bandwidth, we compare the performance of four popular collective communication patterns between MVAPICH2-X-AWS and OpenMPI libraries. In order to analyze the performance in different scales, we repeat the collective performance comparison on 1 node, 4 nodes and 32 nodes scale respectively. Figure 3 shows the single node collective performance comparison between MVAPICH2-X-AWS and OpenMPI. In figure 3(a), MVAPICH2-X-AWS has up to 5.3x lower Allgather latency in small and medium message sizes. In figure 3(b), MVAPICH2-X-AWS is up to 6.1x faster than OpenMPI for small messages, and OpenMPI shows up to 47% lower allreduce latency for messages larger than 128 KB and smaller than 512 KB. In figure 3(c), MVAPICH2-X-AWS shows up to 4.8x lower latency for small and medium message sizes. In figure 3(d), MVAPICH2-X-AWS shows up to 6.7x lower latency on large messages. Through those four collective communication patterns, we can observe that due to different communication patterns, the performance of different MPI libraries diverge or converge in different message sizes. That is because our performance tuning and algorithm selection make effect in different message sizes respectively on those four collective communication patterns. Similarly, figure 4, figure 5 and figure 6 show the performance comparison of the same set of collective communication patterns with different number of nodes. In the experiments of 4 nodes small scale, we observe MVAPICH2-X-AWS has 1.4x lower allgather latency for small and medium message sizes, up to 2.5x lower allreduce latency with small message sizes, up to 4.2x lower gather latency with small message sizes, and 3.9x lower scatter latency with large message sizes. In the



(a) inter-node latency (b) inter-node bandwidth (c) inter-node bi-bandwidth

Fig. 2: Inter-node Point-to-point communication performance of OMB small message latency and large message bandwidth tests on AWS c6gn instances with EFA.



(a) Allgather (b) Allreduce

(c) Gather (d) Scatter

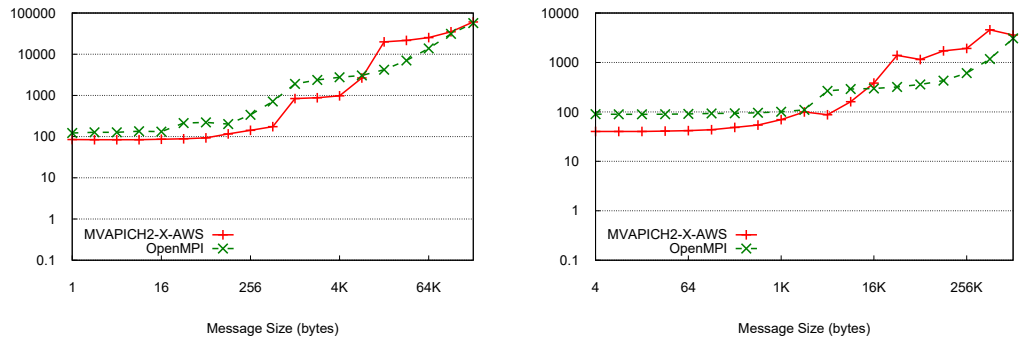
Fig. 3: Performance of OMB collectives with full subscription (64 ppn) on 1 AWS Arm instance with EFA

experiments of 32 nodes large scale allgather and allreduce communication patterns, MVAPICH2-X-AWS still shows up to 1.3x lower allgather and allreduce latency for small and medium message size, while OpenMPI has advantage of up to 2.2x lower latency for large message sizes. In 32 nodes scale gather experiments, MVAPICH2-X-AWS shows 6.2x lower latency for small and medium messages, which is larger than small scale test result. However for 32 nodes scale scatter experiment, the performance gap become smaller to 2.1x lower latency on MVAPICH2-X-AWS.

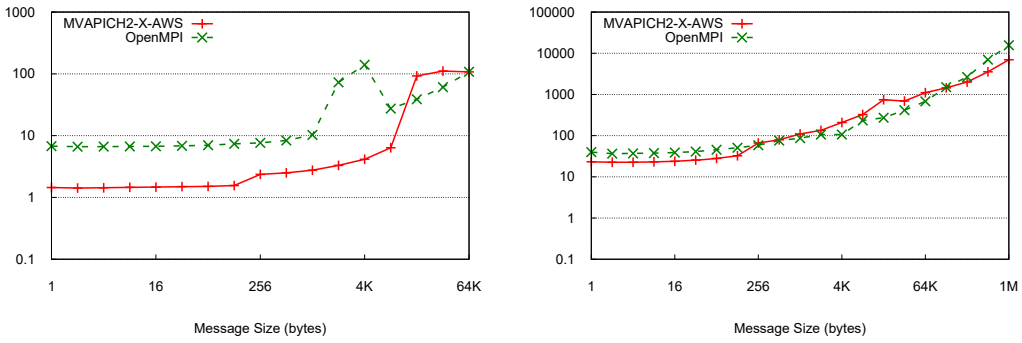
C. Application Results

In addition to micro-benchmark level performance evaluation, we evaluate application level performance with two representative HPC applications – the Weather Research and Forecasting [8] (WRF) model package and Adaptive Mesh Refinement Mini-App [9] (miniAMR).

Figure 7(a) shows the performance comparison between MVAPICH2-X-AWS and OpenMPI with WRF application. For this experiment we use strong scaling input dataset from 12km resolution case over the Continental U.S. (CONUS)



(a) Allgather (b) Allreduce



(c) Gather (d) Scatter

Fig. 4: Performance of OMB collectives with full subscription (64 ppn) on 4 AWS Arm instances with EFA

domain. We run the application with full subscription. As the figure showing, MVAPICH2-X-AWS is more scalable than OpenMPI. On 32 nodes scale, MVAPICH2-X-AWS has 31% less execution time than OpenMPI. Based on our profiling analysis, our WRF experiment execution has scatter and bcast with small message as its main communication pattern. And as it scales out, there are more communication operations involved and we are able to observe more advantage in large scale tests. The optmized MVAPICH2-X-AWS has 5% less execution time than unoptimized version.

Similarly, Figure 7(b) shows the performance evaluation of miniAMR application. MiniAMR is a mini app which applies a stencil calculation on a unit cube computational domain. In performance comparison of miniAMR, we observe MVAPICH2-X-AWS has up to 35% faster execution time than OpenMPI. The optimized MVAPICH2-X-AWS has 3% less execution time than unoptimized version.

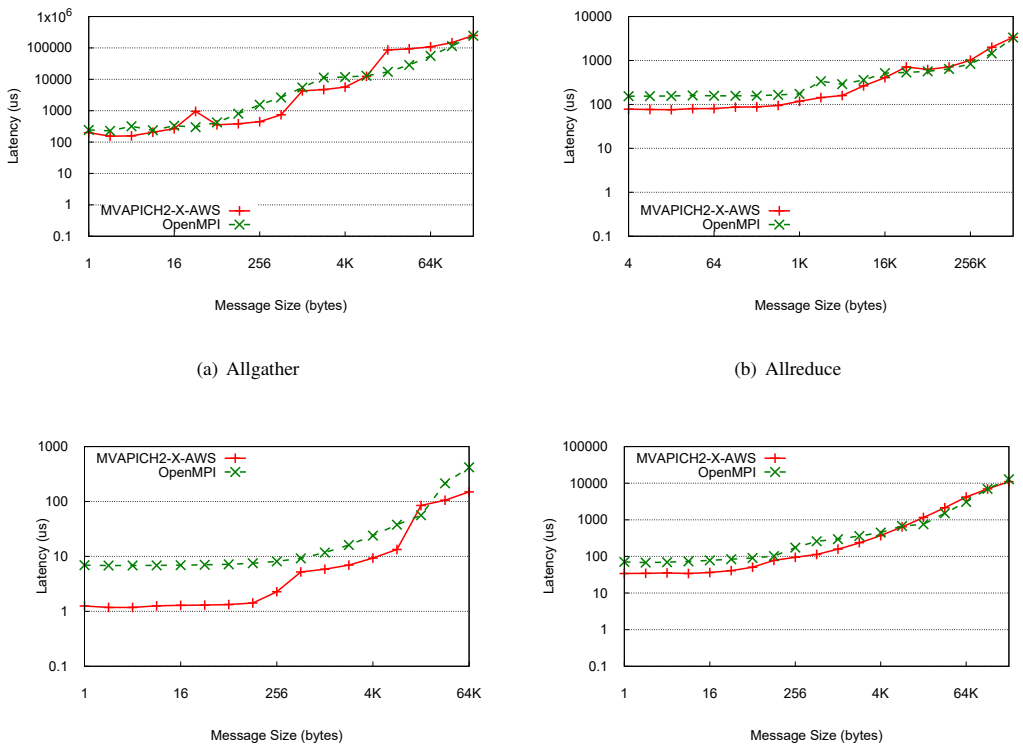
V. Related Work

There are some researchers that have explored HPC application performance on cloud system before. [10]–[14]. Many of them performed experiments on AWS EC2. However, they use non-HPC instances of Amazon EC2 as their experiments were conducted before the availability of Amazon's Cluster Compute instance.

There are some more recent works in this area, they are [15]–[21]. S. Chakraborty et al. [15] propose a new zero-copy design for MPI library to improve communication performance on Amazon EFA and evaluate the impact of SRD transport. This design is based on EFA with "send with immediate" operation disabled. The proposed design provide initial support for MPI libraries to directly transmit messages through scalable reliable datagram. S. Xu et al. [16] proposed an enhanced zero-copy design based on the design proposed in [15]. The enhanced design utilizes "send_with_immediate" operation to straightforwardly reorder out-of-order data with recorded sequence number on receiver's buffer. This paper also performs a cross-platform performance evaluation and characterization between different high performance cloud platforms including Azure and AWS EC2.

Y. Wang et al. [22] perform the experiments of tuning TensorFlow SSD MobileNet on AWS Arm-based cloud systems with Graviton 2 processors. Similarly, Y. Cody Hao et al. [23] perform AutoTVM tuning experiments across severl different AWS Cloud Instance types including Arm-based instances with Graviton processors and Intel x86 processors.

Several other researchers [17]–[21] explore the impact of virtualization or multi-level parallelism on several high performance cloud platforms including Azure and AWS, but most



(a) Allgather

(b) Allreduce

(c) Gather

(d) Scatter

Fig. 5: Performance of OMB collectives with full subscription (64 ppn) on 16 AWS Arm instances with EFA

of their study is on x86 based systems.

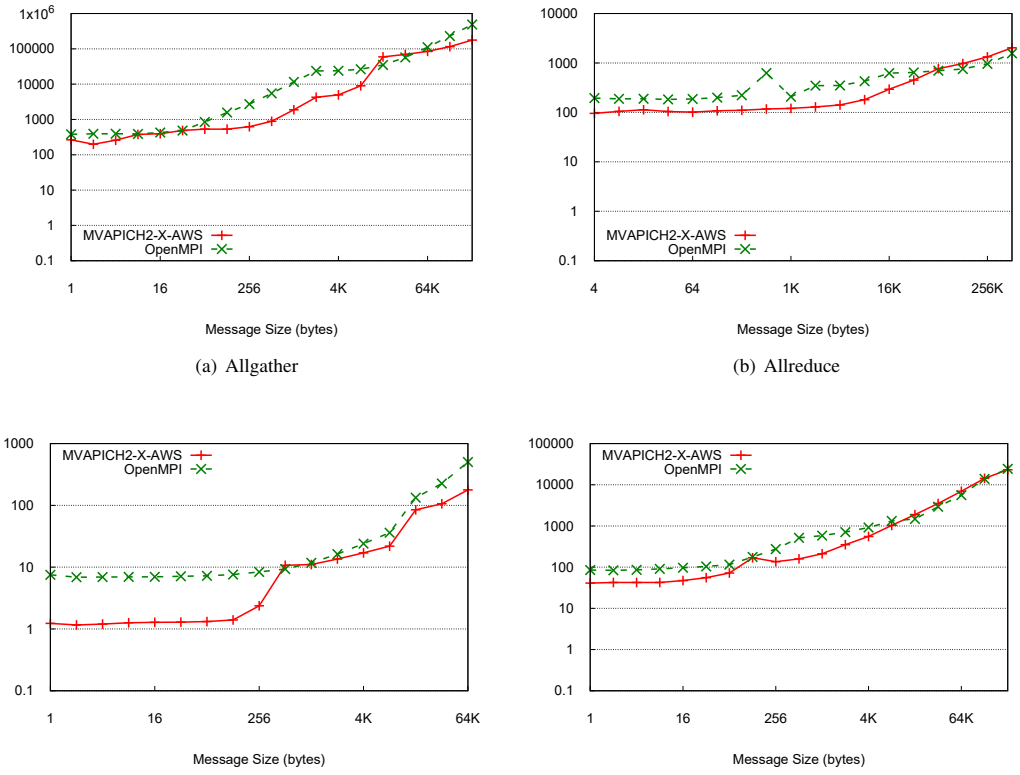
This paper differs from those related study by exploring MPI communication operation performance optimization and comprehensively analyzing and comparing the parallel application performance of different MPI libraries. It brings insights of scientific parallel application optimization on Arm-based cloud systems by leveraging various mechanisms to improve MPI communication performance.

VI. Conclusion

In this paper, we presented our study of performance characterization and optimization of MPI libraries and HPC applications on the emerging Arm-based HPC cloud system of a popular cloud platform, Amazon EC2 using state-of-the-art MPI libraries. With effort of our performance tuning and system customization setting, we contribute to improving large scale collective operation latency by 86% in micro-benchmark level and 9% in application level. In our performance evaluation study, we systematically analyze the performance characteristics of two supported state-of-the-art MPI libraries on AWS Arm-based Cloud system with EFA support. Through our experiments with various parallel HPC applications and middleware on the emerging Arm-based HPC cloud system, we realize that the Arm-based HPC systems has become an unignorable competitor in HPC community with its cost efficiency and competitive performance. We are looking forward to seeing the next Generation Graviton 3 CPU and characterizing MPI and parallel applications on the Arm-based GPU cloud systems.

Acknowledgment

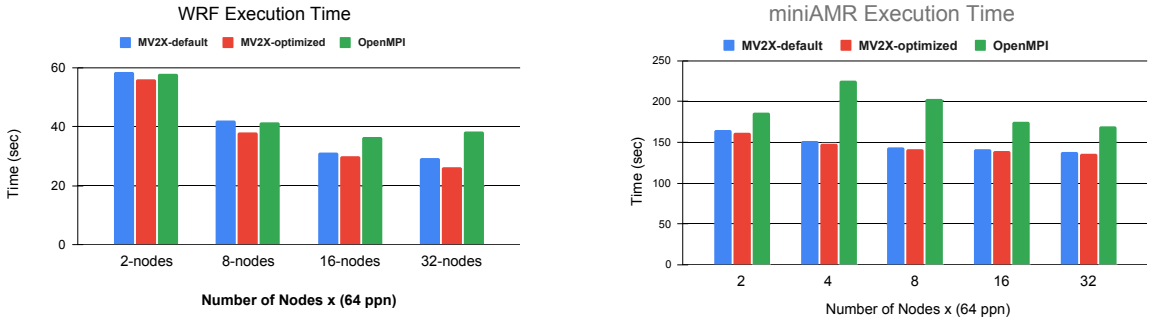
This research is supported in part by NSF grants #1818253, #1854828, #1931537, #2007991, #2018627, #2112606, and XRAC grant #NCR-130002.



(a) Allgather (b) Allreduce

(c) Gather (d) Scatter

Fig. 6: Performance of OMB collectives with full subscription (64 ppn) on 32 AWS Arm instances with EFA



(a) WRF (b) miniAMR

Fig. 7: Two popular HPC application performance comparison on AWS EC2 c6gn instances with full subscription and different number of nodes, comparing unoptimized default MVAPICH2-X-AWS, optmized MVAPICH2-X-AWS and built-in OpenMPI.

REFERENCES

[1] "MPI-4 Standard Document," https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf.

[2] "MPICH: High-Performance Portable MPI," http://www.mpich.org. Accessed: March 21, 2022.

[3] Network-Based Computing Laboratory, "MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE," http://mvapich.cse.ohio-state.edu/.

[4] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, 2004.

[5] "Intel MPI Official Website," https://www.intel.com/content/www/us/en/developer/tools/oneapi/mpi-library.html.

[6] "AWS Parallel Cluster," https://aws.amazon.com/hpc/parallelcluster/.

[7] D. Panda et al., "OSU microbenchmarks v5.6.3," http://mvapich.cse.ohio-state.edu/benchmarks/.

[8] The National Center for Atmospheric Research (NCAR), "Weather Research and Forecasting (WRF) model ," https://github.com/hanschen/WRFV3.

[9] "miniAMR - Adaptive Mesh Refinement Mini-App," https://github.com/Mantevo/miniAMR.

[10] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *2010 IEEE second international conference on cloud computing technology and science*. IEEE, 2010, pp. 159–168.

[11] J. Napper and P. Bientinesi, "Can cloud computing reach the top500?" in *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, 2009, pp. 17–20.

[12] L. Ramakrishnan, R. S. Canon, K. Muriki, I. Sakrejda, and N. J. Wright, "Evaluating interconnect and virtualization performance forhigh performance computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 2, pp. 55–60, 2012.

[13] J. J. Rehr, F. D. Vila, J. P. Gardner, L. Svec, and M. Prange, "Scientific computing in the cloud," *Computing in science & Engineering*, vol. 12, no. 3, pp. 34–43, 2010.

[14] E. Walker, "Benchmarking amazon ec2 for hig-performance scientific computing," *; login:: the magazine of USENIX & SAGE*, vol. 33, no. 5, pp. 18–23, 2008.

[15] S. Chakraborty, S. Xu, H. Subramoni, and D. Panda, "Designing scalable and high-performance mpi libraries on amazon elastic fabric adapter," in *2019 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE, 2019, pp. 40–44.

[16] S. Xu, M. Ghazimirsaeed, J. Hashmi, H. Subramoni, and D. Panda, "Mpi meets cloud: Case study with amazon ec2 and microsoft azure," November 2020.

[17] R. Aljamal, A. El-Mousa, and F. Jubair, "Benchmarking microsoft azure virtual machines for the use of hpc applications," in *2020 11th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2020, pp. 382–387.

[18] P. Smith, S. L. Harrell, A. Younts, and X. Zhu, "Community clusters or the cloud: Continuing cost assessment of on-premises and cloud hpc in higher education," in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, 2019, pp. 1–4.

[19] R. Aljamal, A. El-Mousa, and F. Jubair, "A user perspective overview of the top infrastructure as a service and high performance computing cloud service providers," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. IEEE, 2019, pp. 244–249.

[20] H. A. Hassan, M. S. Kashkoush, M. Azab, and W. M. Sheta, "Impact of using multi-levels of parallelism on hpc applications performance hosted on azure cloud computing," *International Journal of High Performance Computing and Networking*, vol. 13, no. 3, pp. 251–260, 2019.

[21] M. Alfatafta, Z. AlSader, and S. Al-Kiswany, "Cool: A cloud-optimized structure for mpi collective operations," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 746–753.

[22] Y. Wang, X. Zhou, Y. Wang, R. Li, Y. Wu, and V. Sharma, "Tuna: A static analysis approach to optimizing deep neural networks," *arXiv preprint arXiv:2104.14641*, 2021.

[23] C. H. Yu, X. Shi, H. Shen, Z. Chen, M. Li, and Y. Wang, "Lorien: Efficient deep learning workloads delivery," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 18–32.