Enhancing the MPI Collective Communication Performance utilizing iMEX (intelligent Memory EXpander)

ETRI
Supercomputing System Research Section

Hooyoung Ahn

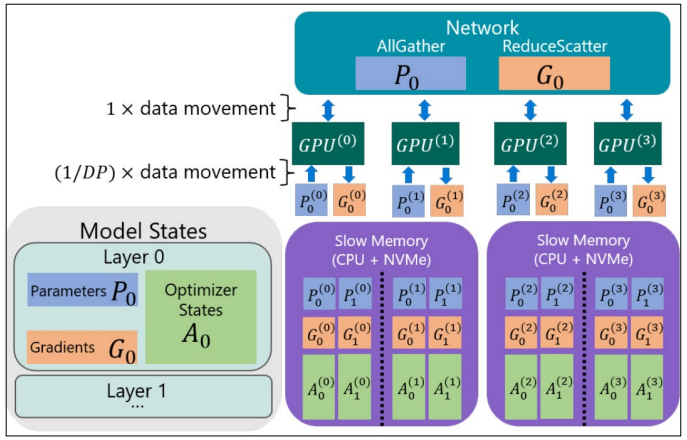
August 21, 2024

Contents

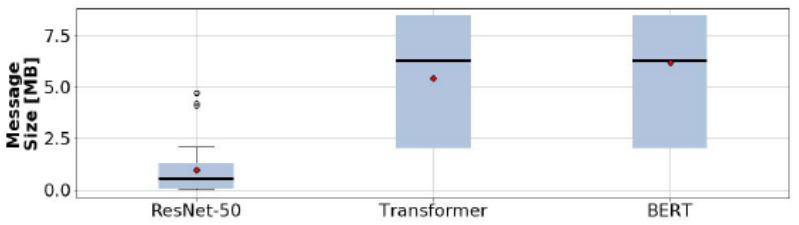
- Motivation & Problem Definition
- Project Goals
- Roles of ETRI and OSU
- Our Approach
- Road Map
- Conclusion

Motivation & Problem Definition (1/3)

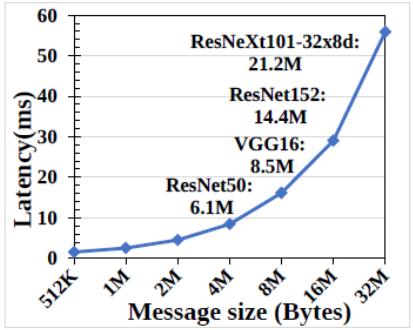
- Large-scale data-intensive applications in HPC and AI require distributed processing in a multi-node environment
  - At this time, there is large and complex communication between nodes, and providing sufficient memory capacity for these applications is one of the necessary conditions for improving performance.
- For example, LLM applications perform distributed training because the huge size of models and training data [1]
  - AllGather and ReduceScatter are used as the main collective communications
  - As the data and model size increases, the collective communication message size increases [2]
  - However, AllGather and ReduceScatter have problems with increased latency for large messages [3]



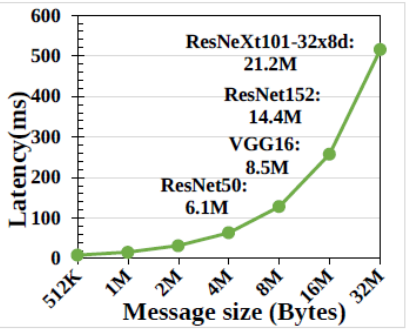
A snapshot of ZeRO-Infinity training [1]



Message Size Distribution for various networks [2]



(a) Allgather latency (16 GPUs)

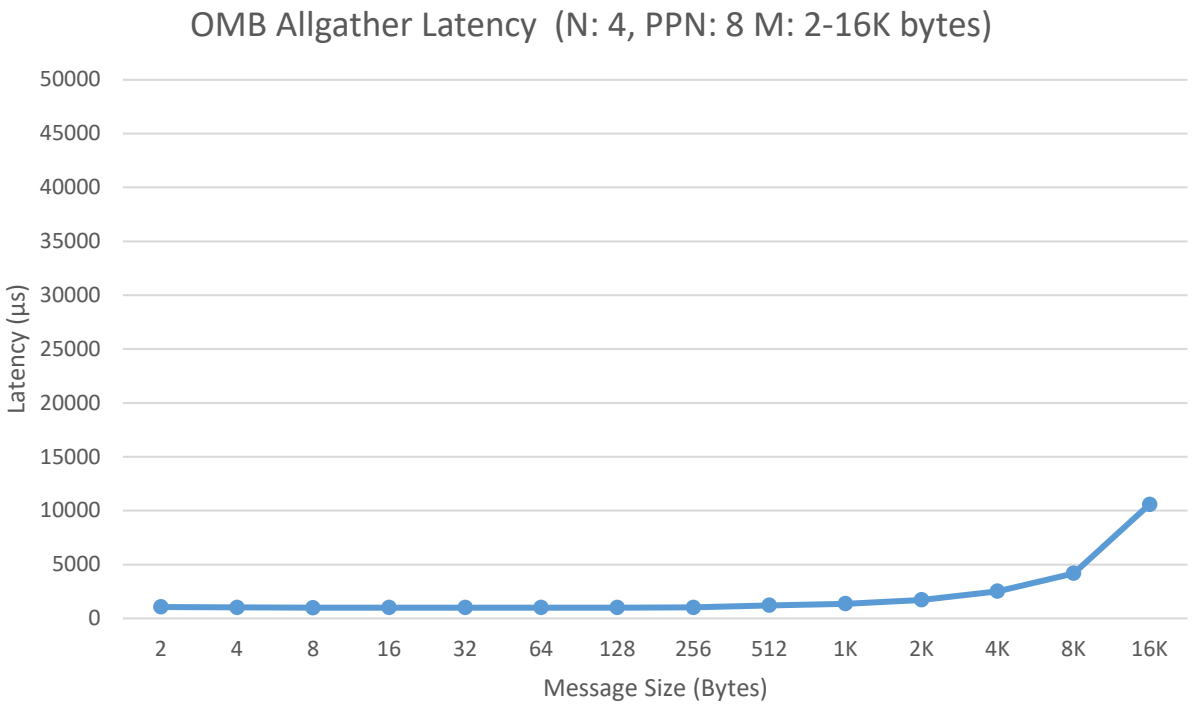


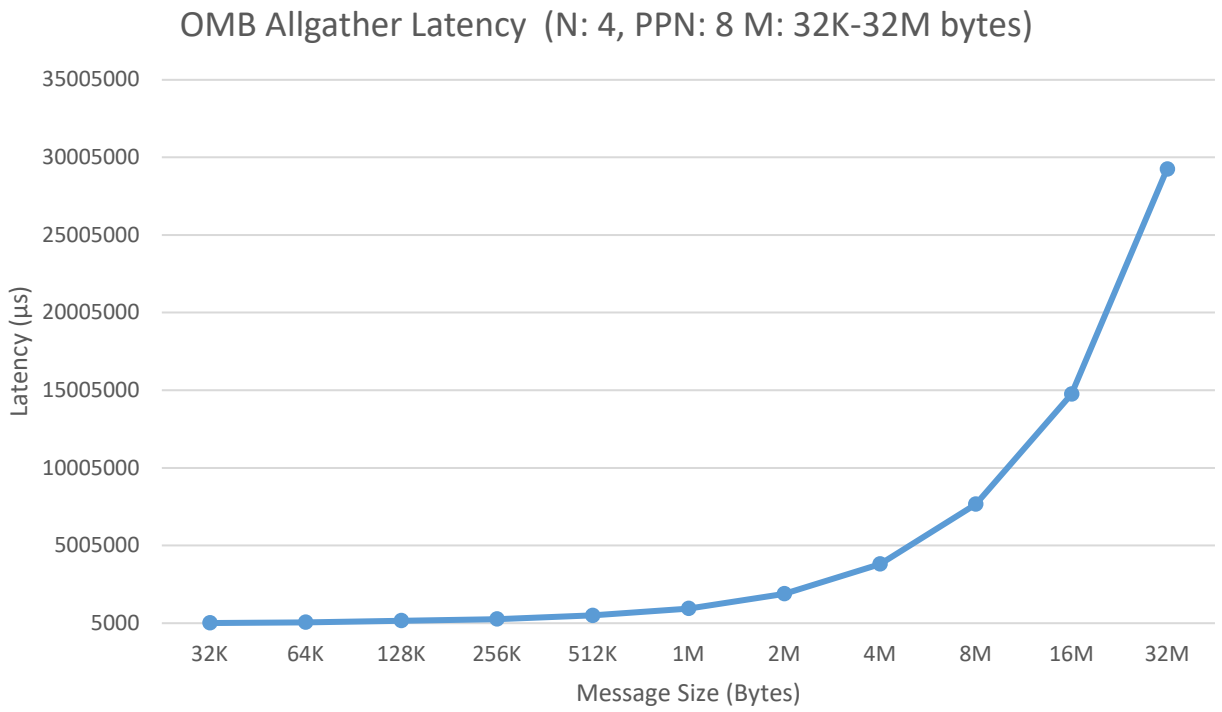
(b) Reduce-Scatter latency

Message sizes of Allgather and Reduce-Scatter in PyTorch FSDP Training on 16 GPUs [3]

Motivation & Problem Definition (2/3)

- As the message size increases, communication latency of traditional allgather also increases

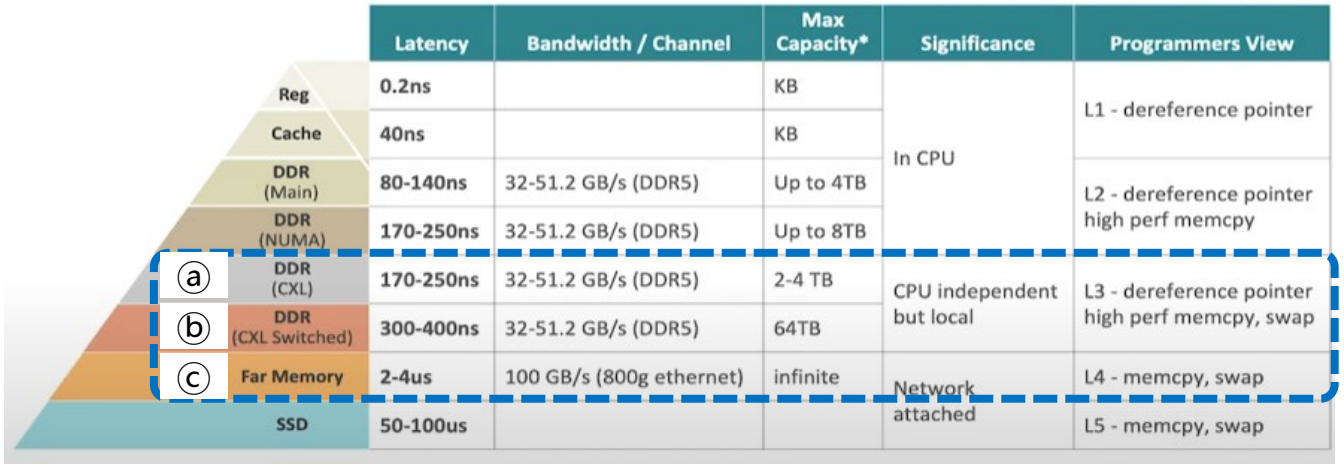




Experimental Results on ETRI's QEMU-based 4 Computing Nodes

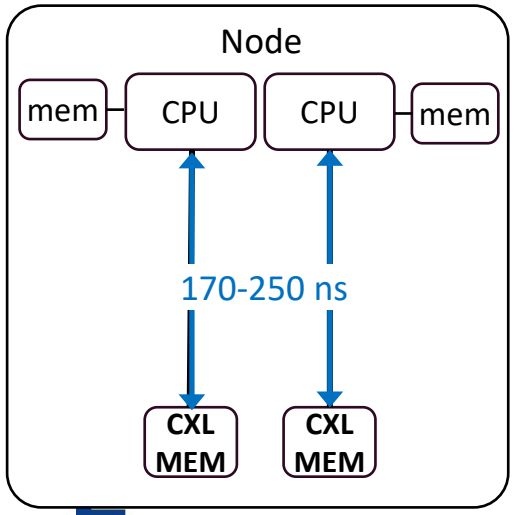
Motivation & Problem Definition (3/3)

- We believed that we could address this issue by using the **CXL interconnect** and the **CXL shared memory pool device in a single rack**, which provide faster communication latency compared to traditional multi-node interconnects using Ethernet or InfiniBand.

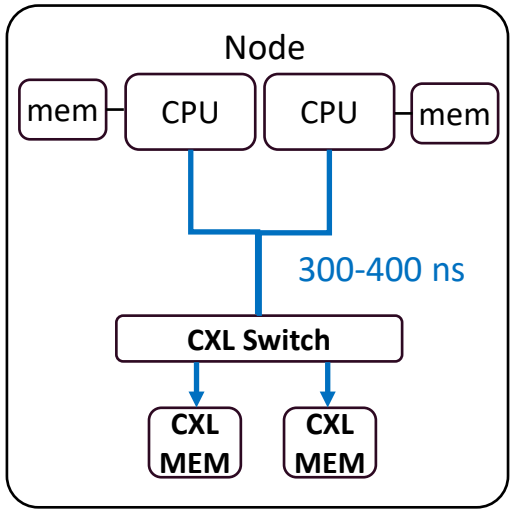


	Latency	Bandwidth / Channel	Max Capacity*	Significance	Programmers View
Reg	0.2ns		KB	In CPU	L1 - dereference pointer
Cache	40ns		KB		
DDR (Main)	80-140ns	32-51.2 GB/s (DDR5)	Up to 4TB		L2 - dereference pointer high perf memcpy
DDR (NUMA)	170-250ns	32-51.2 GB/s (DDR5)	Up to 8TB		
ⓐ DDR (CXL)	170-250ns	32-51.2 GB/s (DDR5)	2-4 TB	CPU independent but local	L3 - dereference pointer high perf memcpy, swap
ⓑ DDR (CXL Switched)	300-400ns	32-51.2 GB/s (DDR5)	64TB		
ⓒ Far Memory	2-4us	100 GB/s (800g ethernet)	infinite	Network attached	L4 - memcpy, swap
SSD	50-100us				L5 - memcpy, swap

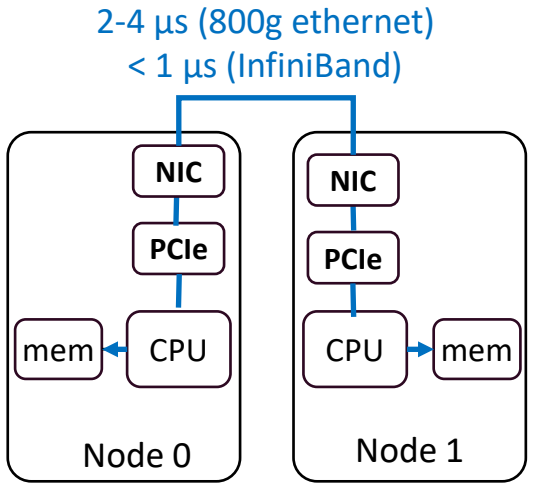
[4]



ⓐ DDR (CXL)

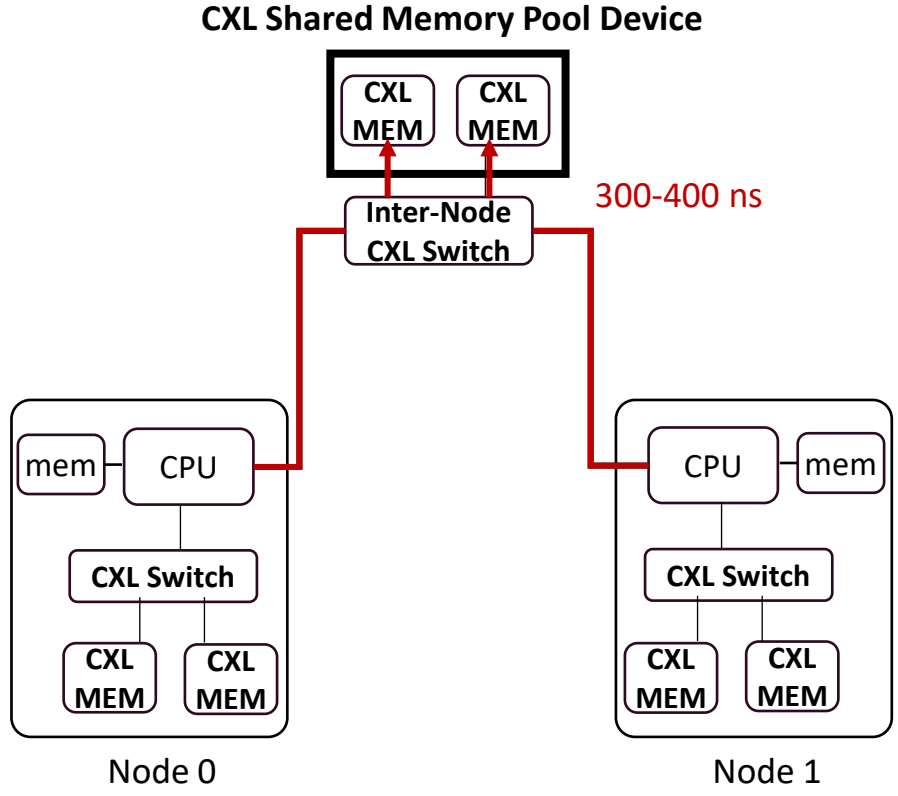


ⓑ DDR (CXL Switched)



ⓒ Network-attached far memory

Because the memory access latency for CXL-attached far memory across nodes can be the same as the latency for CXL-switched memory within a single node, which is about three times faster than the latency of the InfiniBand interconnect.



(Proposed Architecture) CXL-attached far memory

Project Goals

- The goal of this study is to **enhance the MPI Inter-Node collective communication performance** in a **multi-node environment connected by CXL**

- Two Specific Goals
  - **Goal 1**. Utilizing the **CXL shared memory pool** for **collective communication**
    → 1st phase: Sept. 2023 - Aug. 2024
  - **Goal 2**. Utilizing the **intelligent CXL switch** for **collective communication**
    → 2nd phase: Sept. 2024 - Aug. 2025

→ To achieve above goals, we proposed iMEX (intelligent Memory EXpander)

Project Goals

Main memory (512GB) | N Computing Node | : Data Movement
CXL memory (512GB) | MP Memory Pool | n : # of Processes

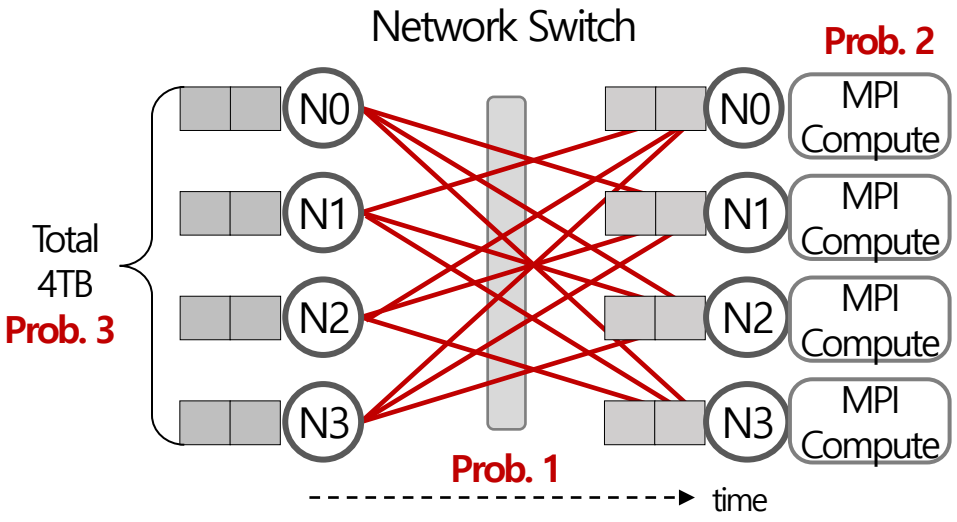
- Key Concept of iMEX

Data-Intensive Applications of **AI** and **HPC** fields

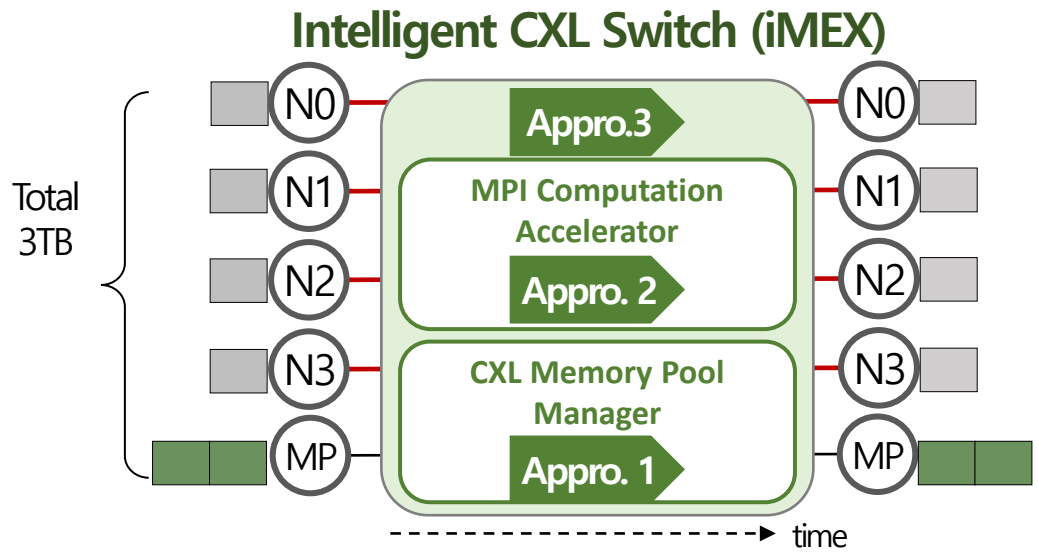
※ **Appro.3. MVAPICH2 optimized for IMEX**

All-Reduce

(AS-IS) Conventional Collective Communication



(TO-BE) Proposed Collective Communication



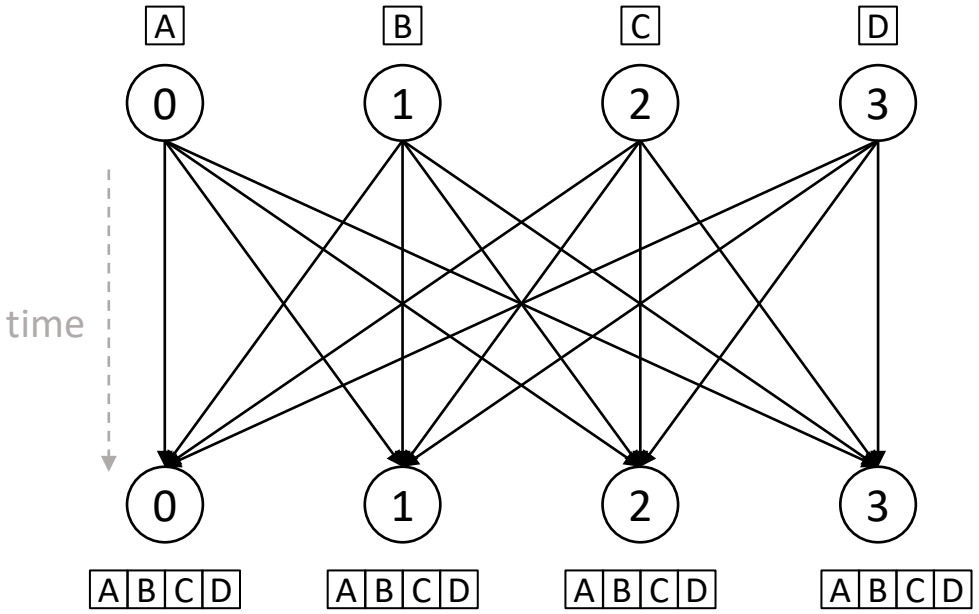
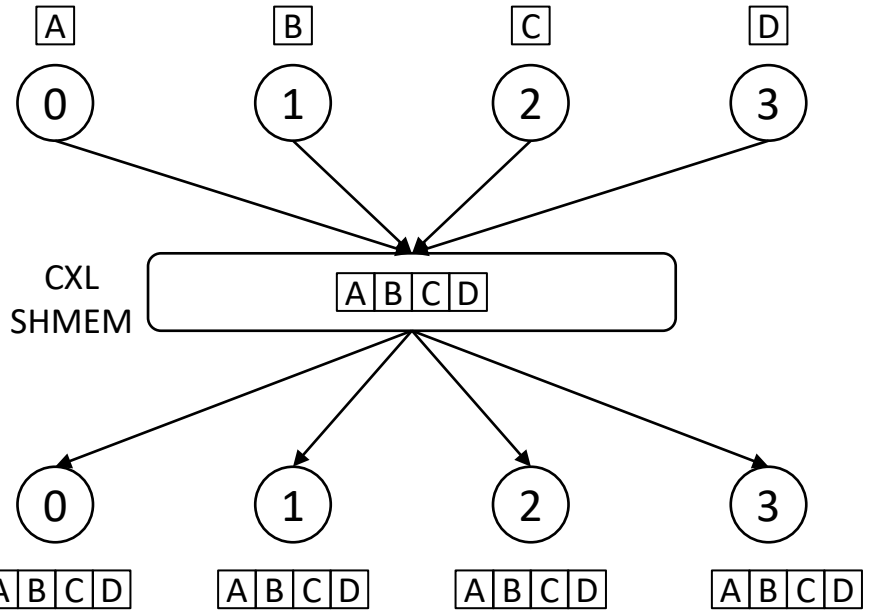
• # of Communication : n^2 • Communication **Latency** : 2-4 μs (100 GE)	Communication	• # of Communication : n • Communication **Latency** : 300-400 ns (CXL Switched DDR)
• MPI Computation on **CPU** • # of Computation : n	Computation	• MPI Computation on **Dedicated Accelerator** • # of Computation : **1**
• **Low**	Memory Utilization	• **High**

Roles of ETRI and OSU

	Research Area	Focus		Research Item
OSU	**Goal 1**	Beyond Rack-Scale CXL Memory Pool	1	Improving collective communication performance by utilizing the **beyond rack scale CXL memory pool device**
			2	Identify and **develop promising demonstration applications** to showcase the CXL-based collective communication proposed in OSU's research item 1
ETRI	**Goal 1**	Single Rack-Scale CXL Memory Pool	1	**Proposed Approach 1**. CXL SHM-based AllGather
	Goal 2	Intelligent CXL Switch	2	**Proposed Approach 2.** In-CXL Switch ReduceScatter

Proposed Approach for Goal 1

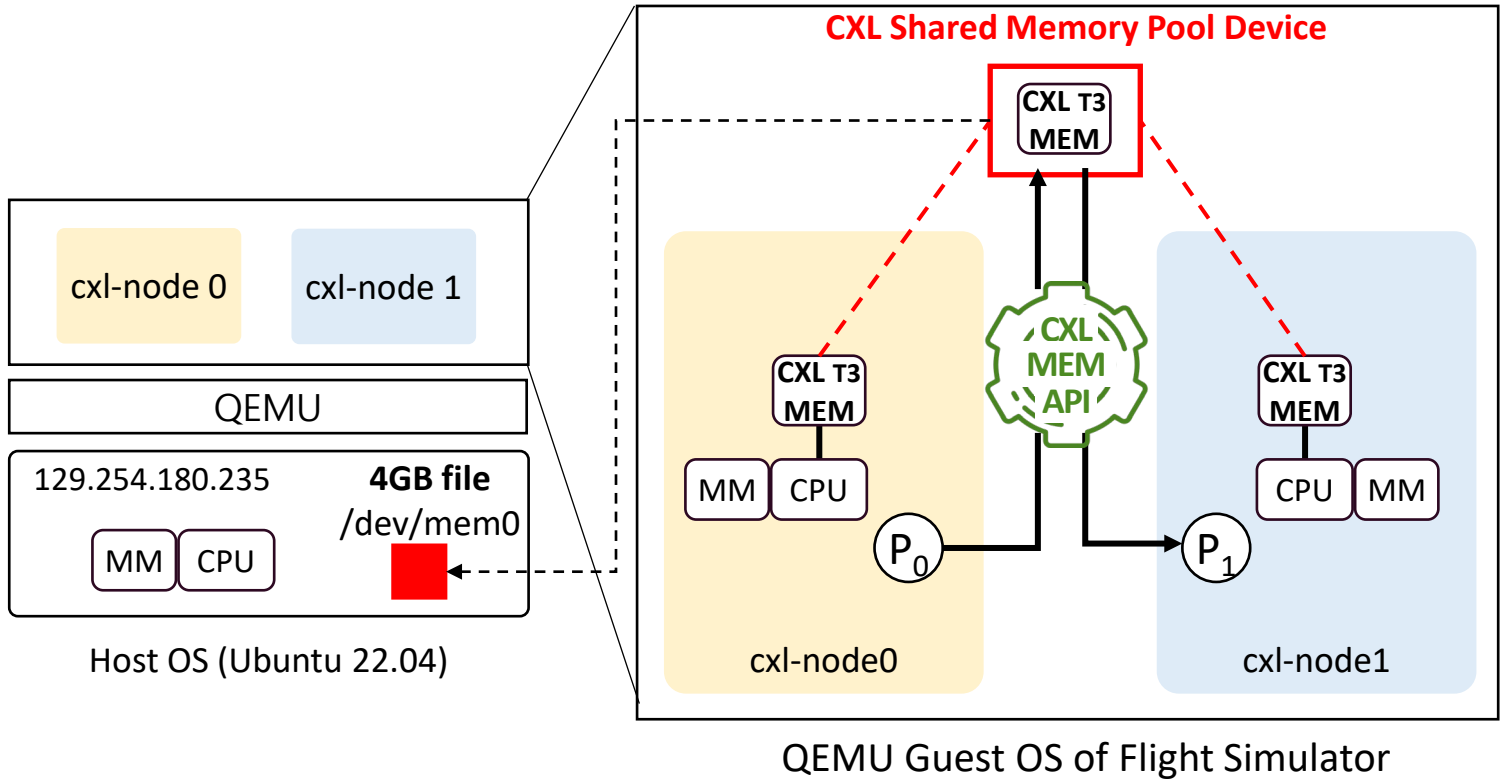
- CXL SHM-based AllGather
  - Design and implement AllGather utilizing the CXL shared memory pool as the collective communication buffer
  - Measure Allgather latency with OMB for performance validation

Conventional. Network-based AllGather	Proposed. CXL SHM-based AllGather
	

Expect performance improvement by
- reducing the number of communications
- achieving performance gains with CXL

Implementation for CXL SHM-based AllGather

- We developed five CXL memory APIs that are utilized for the CXL SHM-based allgather
  - MPI ranks running on different computing nodes can utilize the CXL shared memory pool device as the communication buffer for collective communication



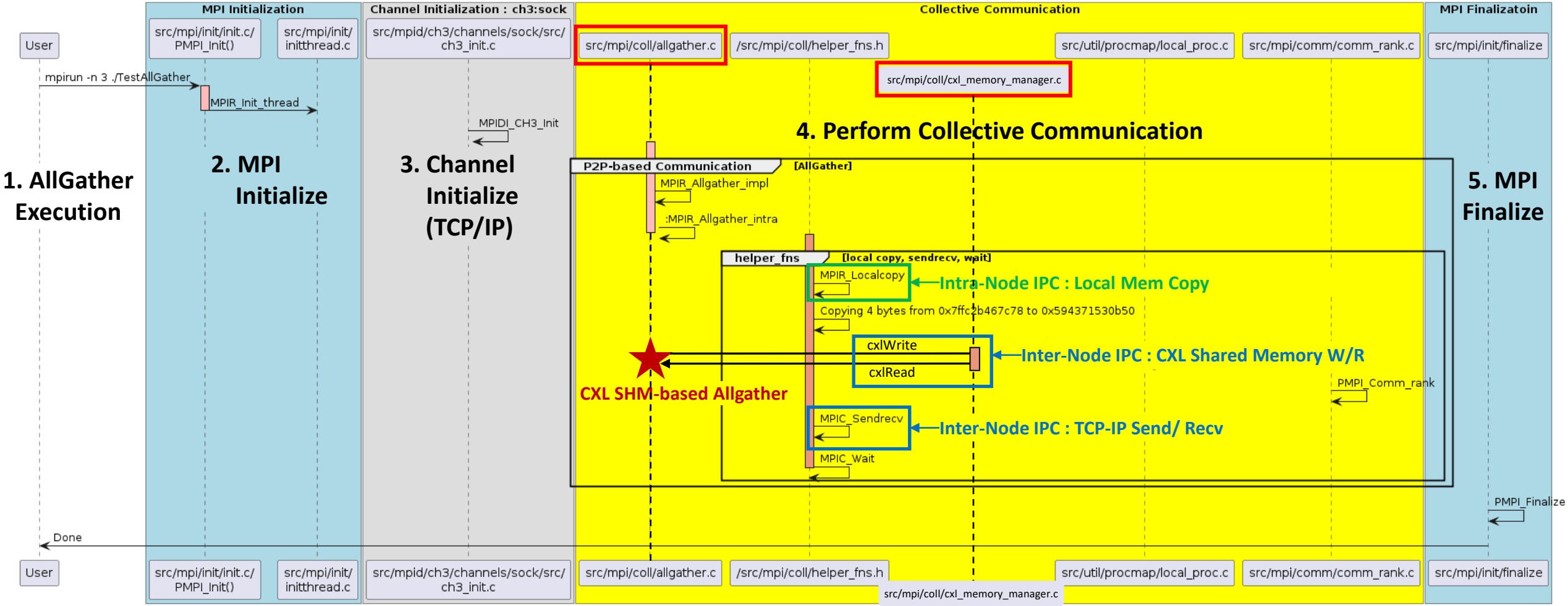
QEMU Guest OS of Flight Simulator

CXL Memory APIs

① cxlmemoryInitialize ()
② cxlmalloc (size)
③ cxlwrite (data, size, offset)
④ cxlread (data, size, offset)
⑤ cxlfree ()

Implementation for CXL SHM-based AllGather

- We implemented the CXL SHM-based allgather in the allgather.c file of MVAPICH2 2.3.7
- We implemented the cxl_memory_manager.c in the coll directory and cxl_memory_manager.h in the include directory



Experimental Setup for CXL SHM-based AllGather

- Software emulator
  - Flight Simulator [5], which emulates the Multi-Node CXL Shared Memory Pool Device in QEMU
- Experimental Environment
  - Host Machine
    - ✓ CPU : AMD EPYC 9754 128-Core Processor
    - ✓ Main memory : 792 GB
  - Guest Machine
    - ✓ QEMU branch cxl-2024-03-05 [6]
    - ✓ OS : fedora release 38 (kernel version : vmlinuz-6.3.7-200.fc38.x86_64)
- Benchmark Suite
  - OSU Micro Benchmarks [7]

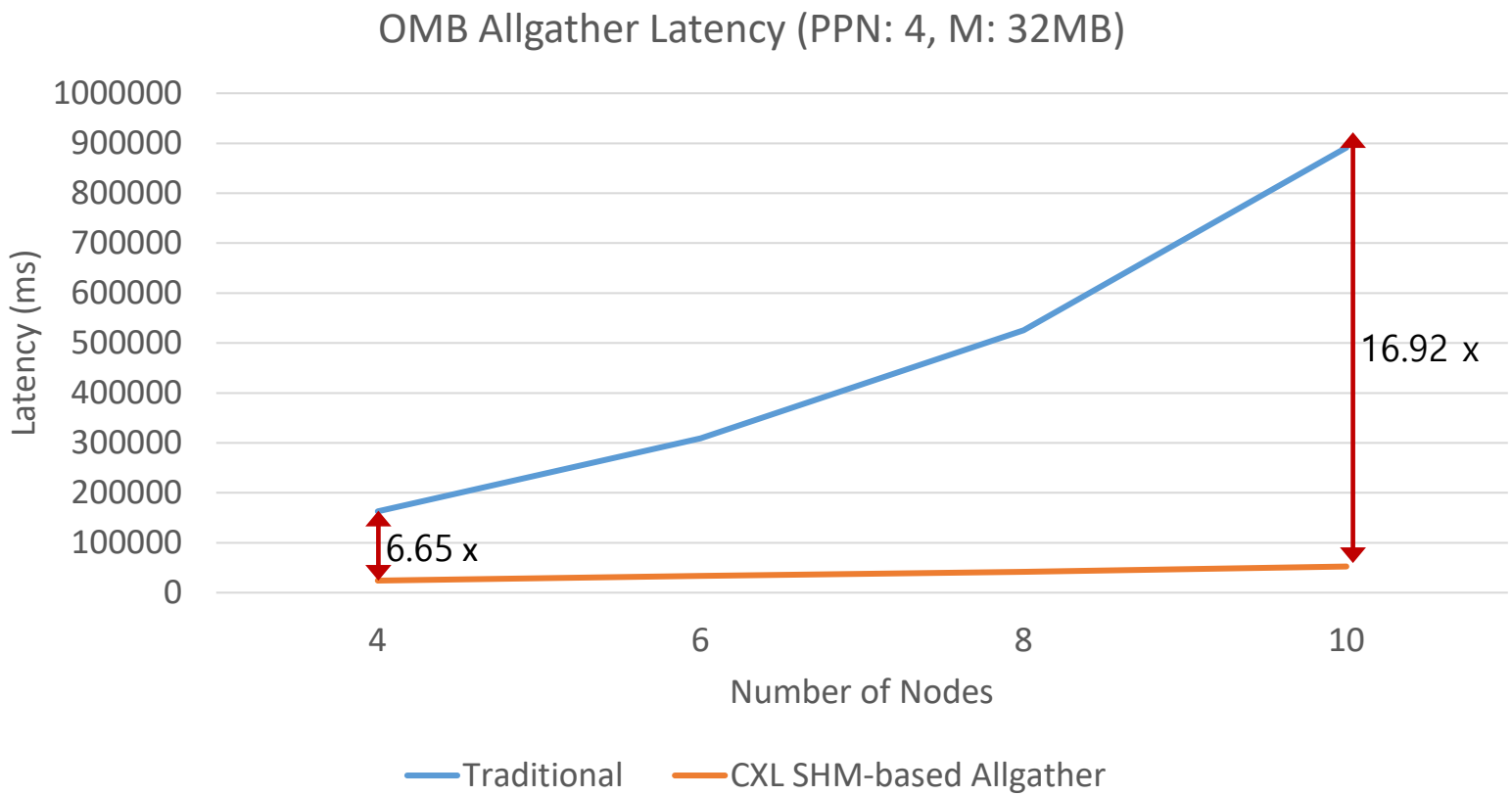
Experimental Items for CXL SHM-based AllGather

	Performance metrics to be measured	Metric (y-axis)	Variable (x-axis)	Fixed Parameters	
1	Performance with increasing number of nodes	OMB AllGather latency	# of nodes (guest OS) (e.g., 2, 4, 8, 16)	1	PPN
				2	message size
2	Performance with increasing PPN		PPN (e.g., 1, 2, 4, 6)	1	# of nodes
				2	message size
3	Performance with increasing message size		message size (e.g., 512KB-32MB)	1	# of nodes
				2	PPN

※ PPN (Process Per Node)

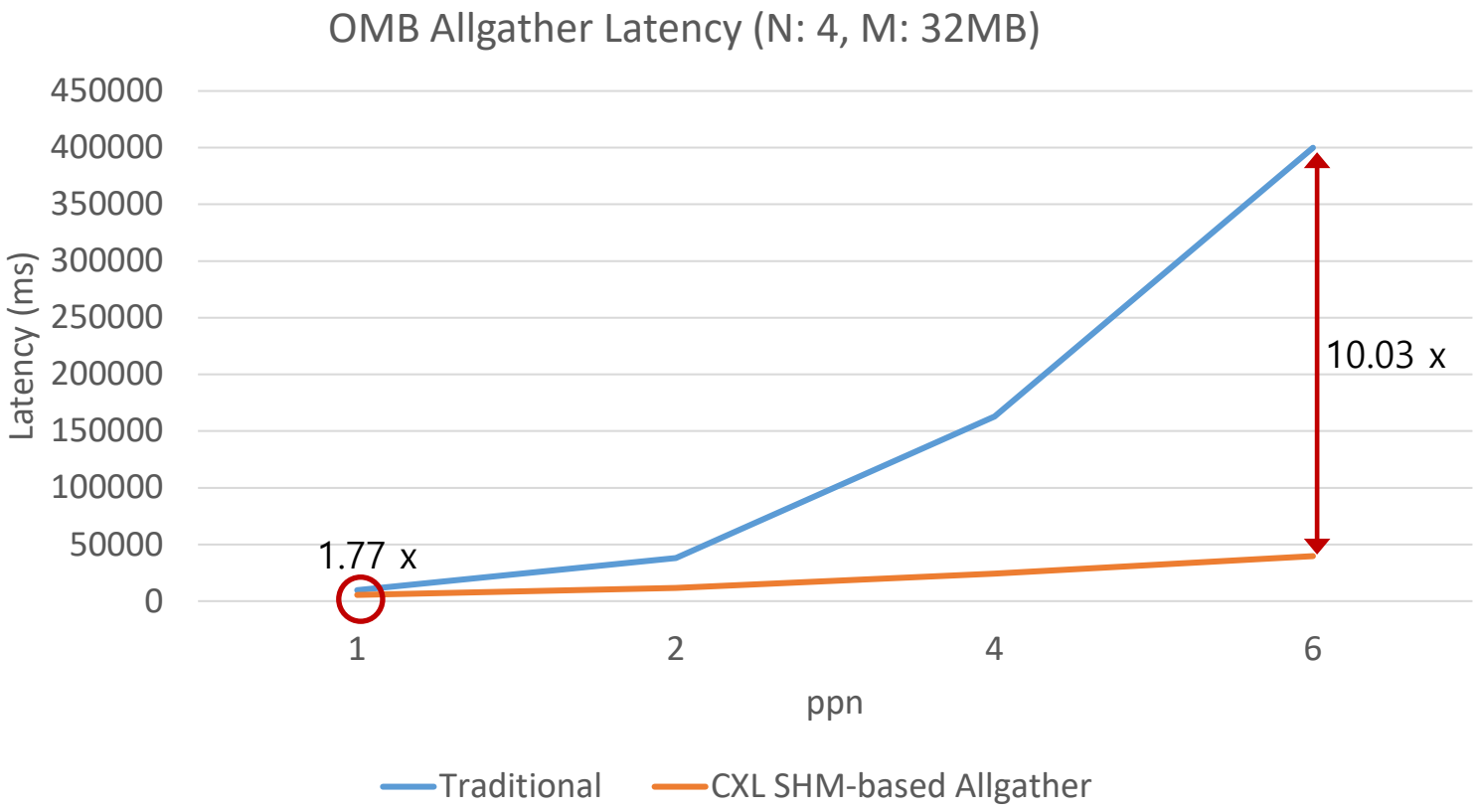
Experimental Results for CXL SHM-based AllGather

- Performance as the number of nodes increases
  - The results showed that with 10 nodes, the maximum performance improvement was 16.92 times
  - With 4 nodes, the minimum performance improvement observed was 6.65 times



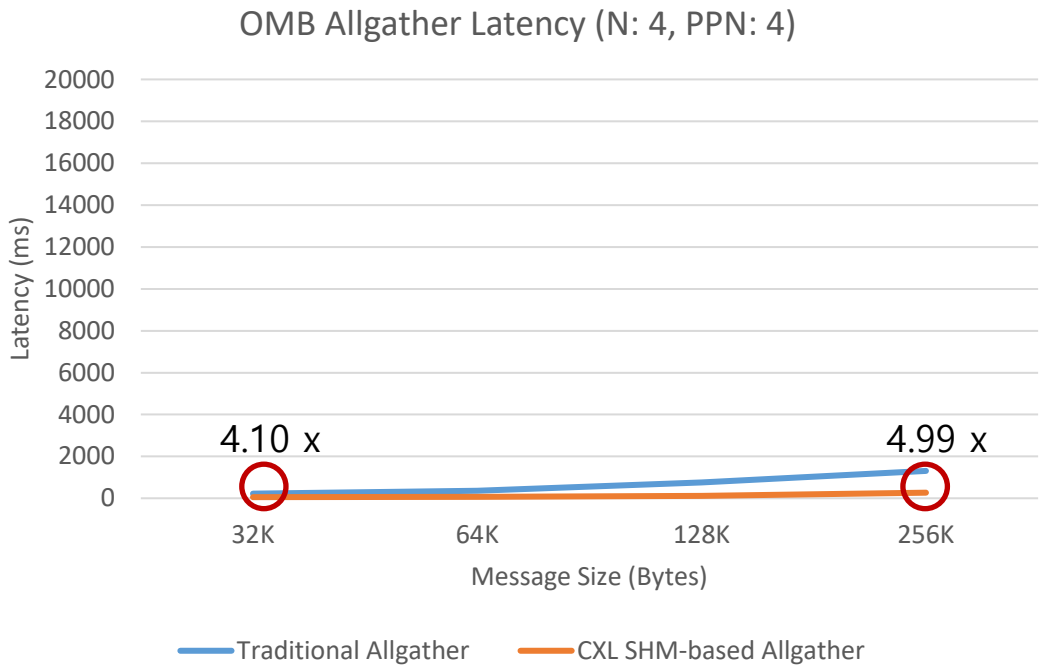
Experimental Results for CXL SHM-based AllGather

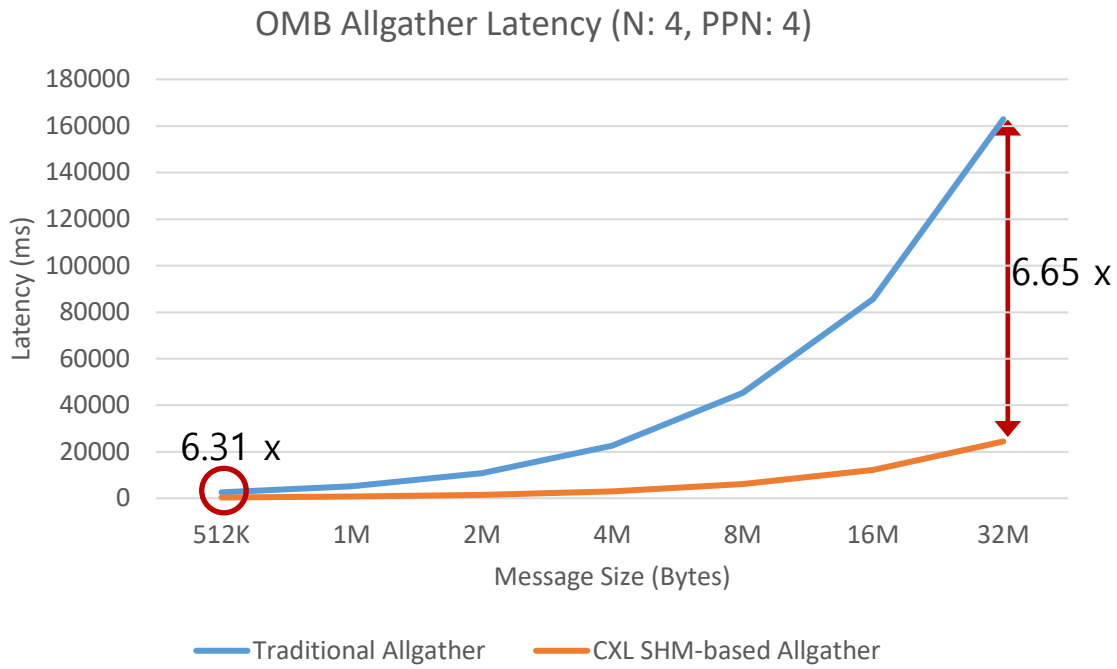
- Performance as the PPN increases
  - The results showed that with 6 PPN, the maximum performance improvement was 10.03 times
  - With 1 PPN, the minimum performance improvement observed was 1.77 times



Experimental Results for CXL SHM-based AllGather

- Performance as the message size increases
  - For mid-sized messages, we achieved a maximum performance improvement of 4.99 times
  - For large-sized messages, we achieved a maximum performance improvement of 6.65 times





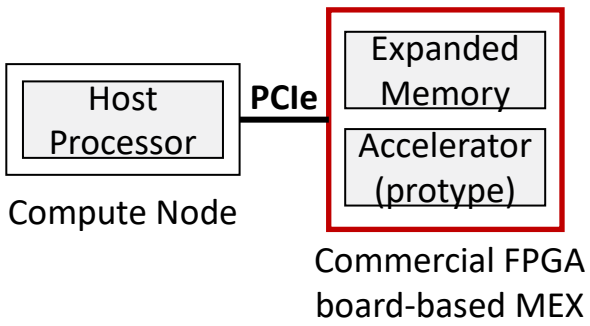
Road Map

- We aim to improve the performance of ***data-intensive applications*** in ***multi-node systems***

Now, we are here

Stage 1. MEX

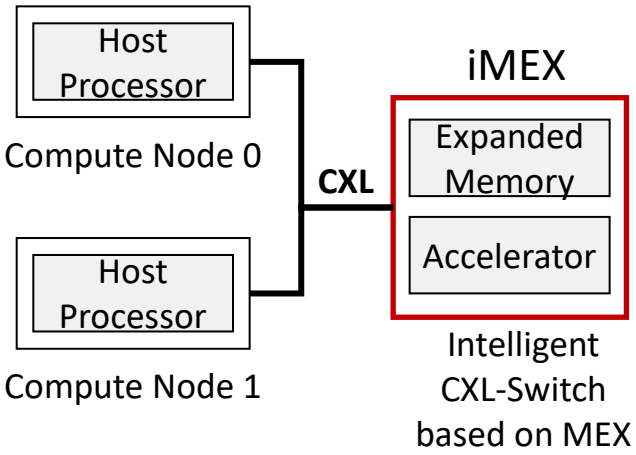
- Commercial FPGA board-based MEX
- Up to 32GB expanded memory
- Prototype version of accelerator
- Support a single node



※ MEX (Memory EXpander)

Stage 2. iMEX

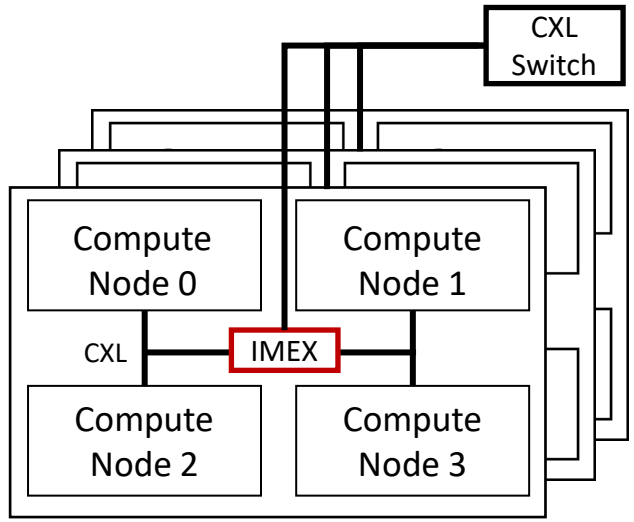
- Support multi-node system using CXL
- Accelerate MPI collective operation using dedicated accelerator
- Use CXL Memory Pool for expanded memory capacity



※ iMEX (intelligent MEX)

Stage 3

- Improvement the scalability of iMEX
- Multiple iMEX devices will be connected to a CXL Switch
- Support more complex topology



Conclusion

- We expect to enhance the **collective communication performance** utilizing **iMEX's MPI Computation Accelerator**
- We expect to Improve the **Memory Utilization** for HPC systems utilizing **CXL Memory Pool** as a MPI Communication buffer
- We expect to Improve the AI and HPC **Application performance** by reducing the Communication Cost
- We plan to showcase the **research progress of iMEX** at **SC24**

References

1. Rajbhandari, Samyam, et al. "Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning." Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021.
2. KLENK, Benjamin, et al. An in-network architecture for accelerating shared-memory multiprocessor collectives. In: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020. p. 996-1009.
3. Zhou, Qinghua, et al. "Accelerating distributed deep learning training with compression assisted allgather and reduce-scatter communication." 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, 2023.
4. "Enfabrica Scaling CXL Memory Using High Speed Networking ", https://www.youtube.com/watch?v=YdJWhqeT5DM
5. “MemVerge Flight simulator, ” https://memverge.com/cxl-qemuemulating-cxl-shared-memory-devices-in-qemu/
6. “QEMU-CXL branch,” https://gitlab.com/jic23/qemu
7. “OSU Micro-Benchmarks,” https://mvapich.cse.ohio-state.edu/benchmarks/

Thank You!

Contacts : ahnhy@etri.re.kr