CXL for AI workloads

ECE/CS598HPN

Radhika Mittal

Acknowledgements: slides made with help from AI (Claude)

Thank you for your feedback!

- What you like: paper organization and discussions.
- Least useful part: paper reviews (multiple of you)
  - Some also said they like it most.
- Suggestions:
  - More background before paper reading.
  - Course project seems to be more stressful for some of you.
  - Some MPs for hands-on experience.
  - Some questions.
- Most liked paper varied. Popular disliked papers: SpectrumX, UE

- *Is handwriting getting worse with newer generations?*

Today's paper: CXL-CCL

- Last two lectures: how to enable GPUs in different servers to effectively communicate over an Ethernet fabric (via RDMA / UltraEthernet).

- Today's paper: can we skip the Ethernet fabric and communicate via a shared memory pool?

- The vehicle: CXL (Compute Express Link) — a PCIe-based coherent-memory interconnect.

Background: intra-host interconnects

- Inter-node communication: Ethernet, Infiniband
- Intra-node communication (e.g. NIC-CPU, GPU-CPU, GPU-GPU):
  - Typically PCIe
  - Recent intra-node GPU-GPU interconnects: NVLink, NVSwitch.

Background: What Is CXL?

Compute Express Link — an open, cache-coherent interconnect over the PCIe physical layer.

CXL.io — device discovery, configuration, management (essentially enhanced PCIe)

CXL.cache — an attached device coherently caches host memory

CXL.mem — the host directly loads/stores memory that physically lives on the device

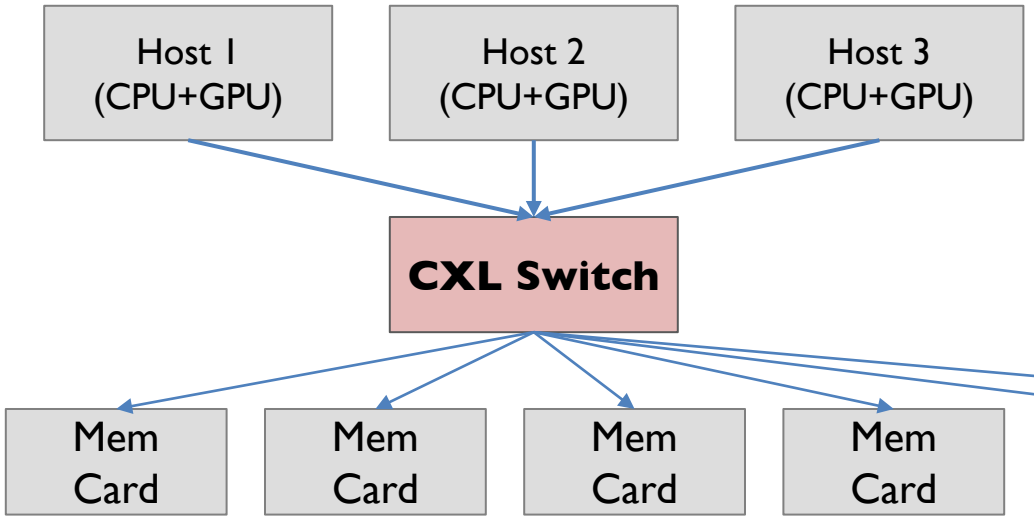
All three protocols are multiplexed over the same physical PCIe link.

Type 1	Type 2	Type 3
Accelerator, no local memory (uses CXL.cache)	Accelerator + local memory, e.g. GPUs (uses CXL.cache + CXL.mem)	Pure memory device — expansion or pooling (uses CXL.mem only)

The paper's CXL memory cards are Type 3 devices — pure memory, no compute.

Background: Memory Pooling & CXL Switches

A single CXL link just extends one host's memory.
A CXL switch lets several hosts share a set of CXL memory devices as a pool.



Each host maps the pool via Device-DAX, then just uses ordinary load/store.

Alleged Limitations of RDMA based transfers

- Extra copy in GPU.
- CPU involvement in control path.

- *GPU Direct RDMA has neither of these limitations.*
  - *HBM directly mapped to GPU.*

- Paper also RDMA based approaches have higher cost, higher software stack complexity, and lower availability.

Challenges with using CXL

- No support for load-balanced data placement in the memory pool.
- Handling semantics of data publication and retrieval.
  - no cache coherence across nodes.

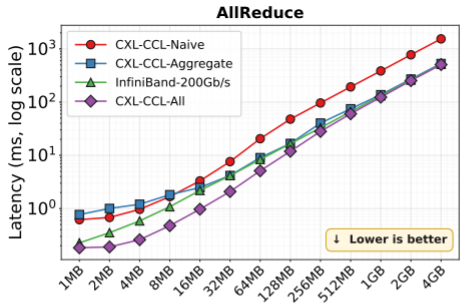
Dealing with the challenges

- CXL-CCL provides library level support for load-balanced placement and retrieval.

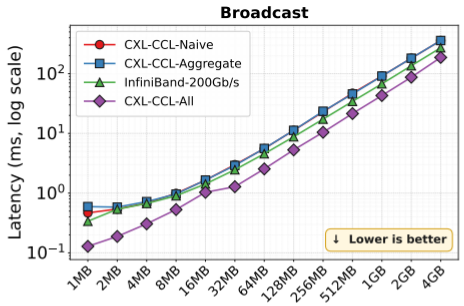
- Locking for semantic correctness.
  - Data split into chunks.
  - Semaphore associated with each chunk.
  - Server process (CPU?) updates semaphore of a chunk once data write from its GPU completes.
  - Another server process polls for change in the semaphore.
  - When state change detected, proceed with reading to GPU memory (doorbell).

Issues with this approach?

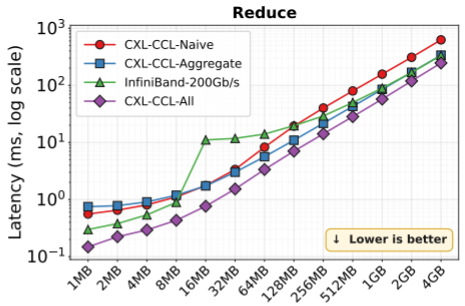
Evaluation



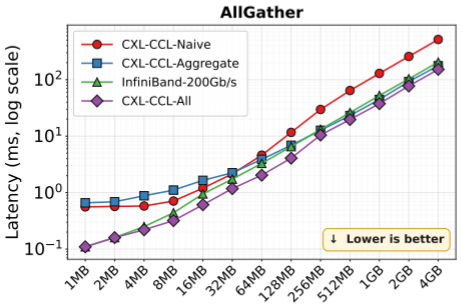
(a) AllReduce.



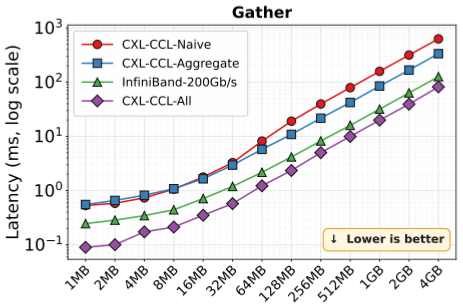
(b) Broadcast.



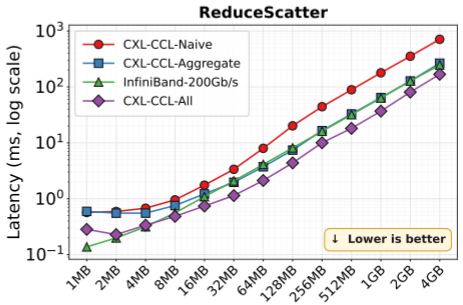
(c) Reduce.



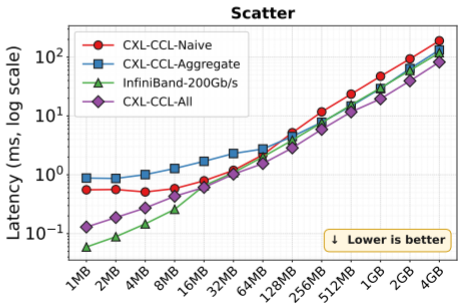
(d) AllGather.



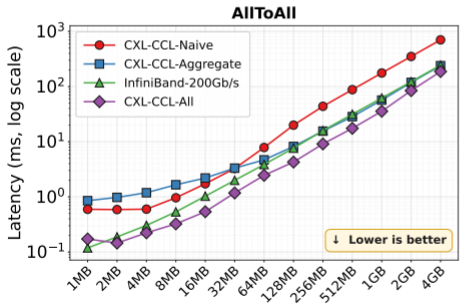
(e) Gather.



(f) ReduceScatter.



(g) Scatter.



(h) AllToAll.

Your thoughts?

Pond: CXL-Based Memory Pooling Systems for Cloud Platforms (ASPLOS'23)

- From Microsoft Azure
- Server memory provisioned for peak usage.
- Can instead provision for average case, and use CXL memory pools during peak.
- Argument: this will be cheaper.
- Primary setting: cloud; VM placement.

Case against CXL memory pooling (HotNets'23)

- From Google.
- Cost: CXL requires a parallel network infrastructure.
- Software complexity of memory management.
- Usecase?
  - Found limited cases where pooling reduces memory stranding for VM packing.