Data Processing on Modern Hardware

Jana Giceva

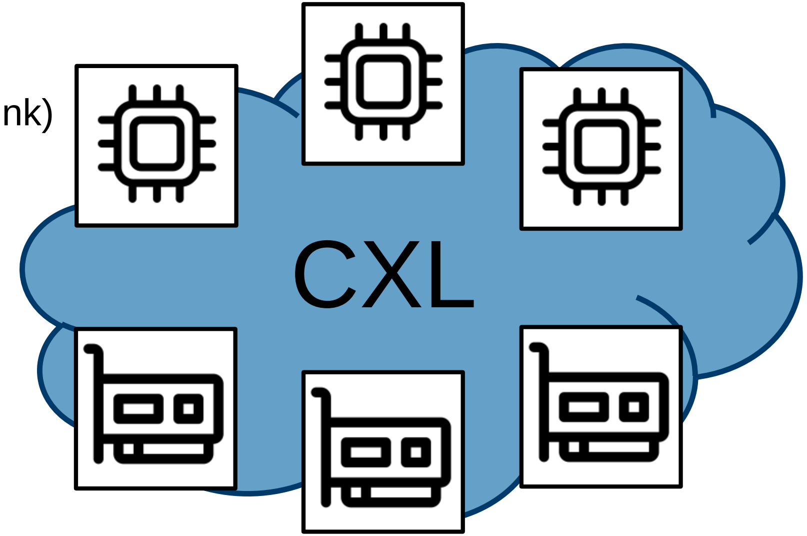
Lecture 11: CXL, memory-centric DBs

Compute Express Link

- Standardized interconnect technology between CPUs and devices
- Allows CPUs and devices to access and cache data stored in each other’s memory

- Maintains cache coherence

- Based on the PCIe (5.0/6.0) physical layer:
  - Offers coherency and memory semantics that scale with the PCIe bandwidth (~63/121 GB/s per x16 link)



Interconnect Limitations (PCIe and DDR)

Limitations – Coherent accesses

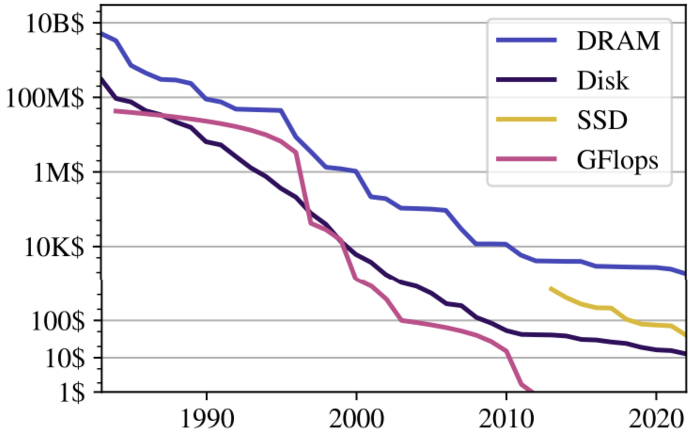
- **Accesses from PCIe devices to system memory**
  - Non-coherent reads/writes
  - PCIe cannot cache system memory to exploit temporal/spatial locality

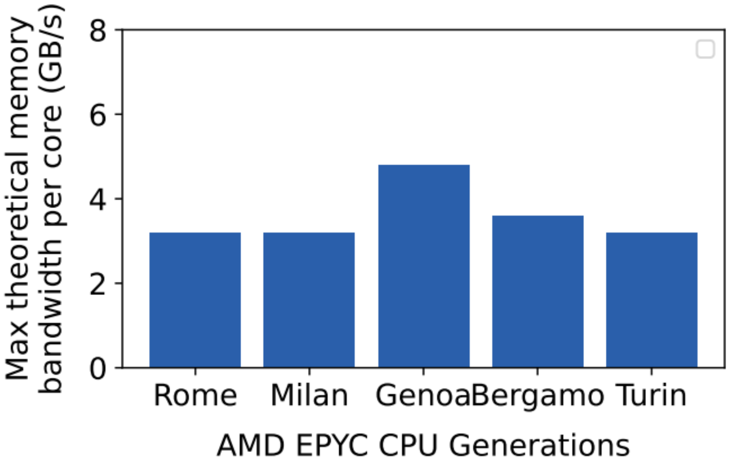
- **A host accesses a PCIe device’s memory non-coherently**
  - Device memory cannot be mapped to the cache-able system address space

- **Utilizing accelerators**
  - Data structures are moved from the host-s main memory to accelerator for data processing before being moved back to main memory
  - Multiple devices cannot access parts of the same data structures simultaneously with the CPU without moving entire data structures back and forth.

Limitations – Memory scalability

- Demand for memory increases proportionate to the exponential growth of compute.
- DDR memory fails to match this demand, limiting memory capacity and bandwidth per CPU



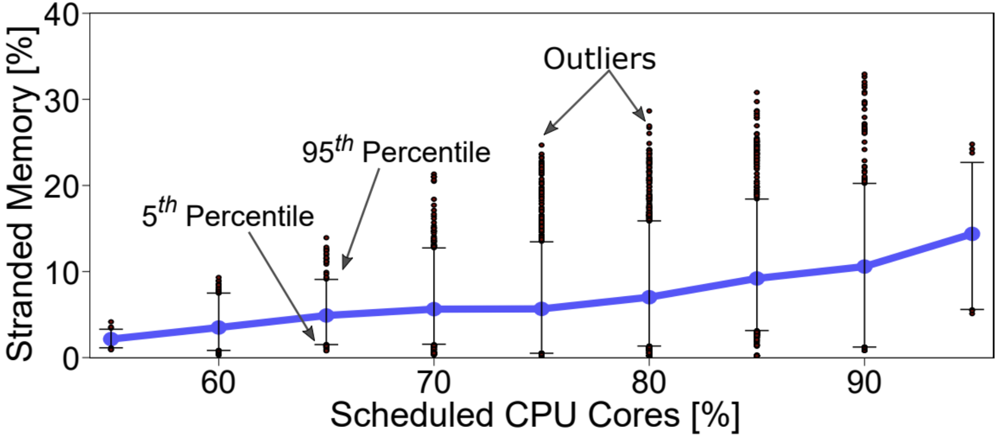


Chronis et al. Databases in the Era of Memory-centric Computing, CIDR’14

Limitations – Stranded resources

- Stranded resource when idle capacity remains while another resource is fully used.
- Cause: tight coupling of different resource types on one motherboard

- Result: servers overprovisions resources to handle workloads with peak capacity demands.



Chronis et al. Databases in the Era of Memory-centric Computing, CIDR'14

Limitations – Data sharing

- Distributed systems often rely on fine-grained synchronization
- Updates are often small and latency sensitive
  - Example: distributed databases with
    - 1 kB sized pages
    - Distributed consensus (e.g., data communicated to agree on what transactions to commit to a database in which order) with even smaller updates.
- With small data chunks, communication delay in typical datacenter network dominates the wait time for updates, slowing down these use-cases.

Limitations of PCIe and DDR

- Non-coherent Memory Access (PCIe)
  - Caching data allows exploiting the temporal and spatial locality of data accesses

- Memory Bandwidth Limitations (DDR)
  - DDR memory bandwidth grows, but doesn’t match the growing number of cores per CPU

- Homogeneous Memory mediate type (DDR)
  - DDR-attached memory must support DRAM-specific commands, lacking adoption of new media types

- CPU-coupled memory (DDR)
  - hinders independent scaling of individual resources

- Memory Sharing (PCIe)
  - PCIe doesn’t support sharing memory across systems while maintaining cache coherent access.

Compute Express Link (CXL)

CXL Technology Basics

- Memory speed-access over PCIe physical layer

- Supports new architectures:
  - Disaggregated memory
  - Pooled memory
  - Switches for memory fabrics
  - Shared memory
  - Persistent memory

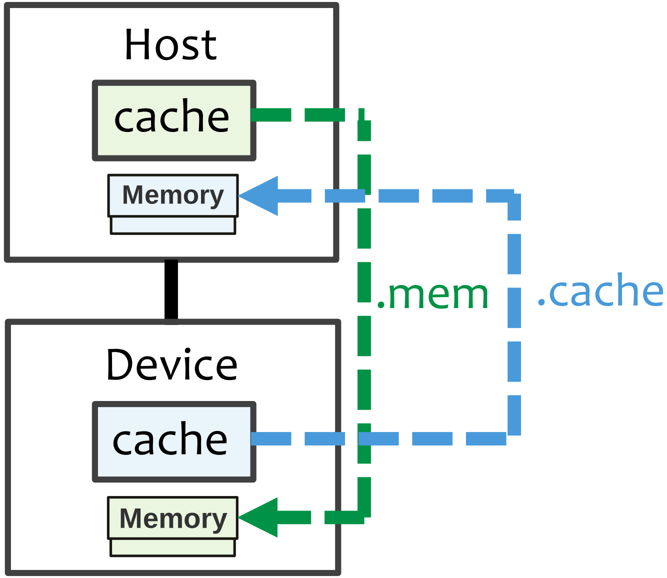
CXL Protocols

- CXL defines three protocols that are negotiated over the PCIe

- **CXL.io**
  - Base protocol: used for e.g., device enumeration, initialization, device registration
  - Based on PCIe 5.0/6.0
  - Non-coherent load-store semantics of PCIe

- **CXL.cache**
  - Devices can *cache* data stored in system memory

- **CXL.mem**
  - CPU can access and cache data stored in CXL device memory



- CXL.io is mandatory to get an endpoint on CXL. CXL.cache and/or CXL.mem are add ons so to speak.

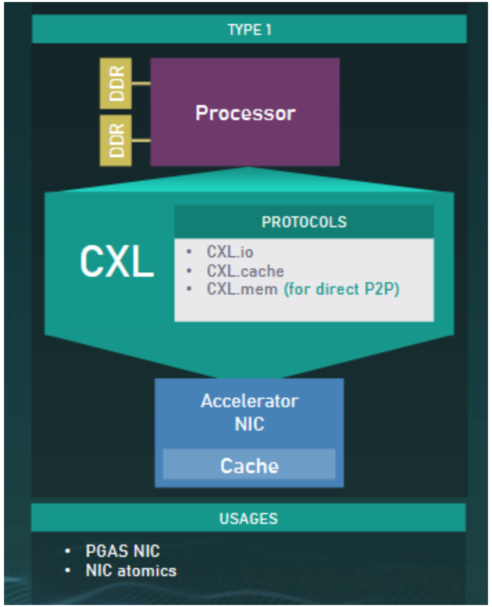
CXL Device types

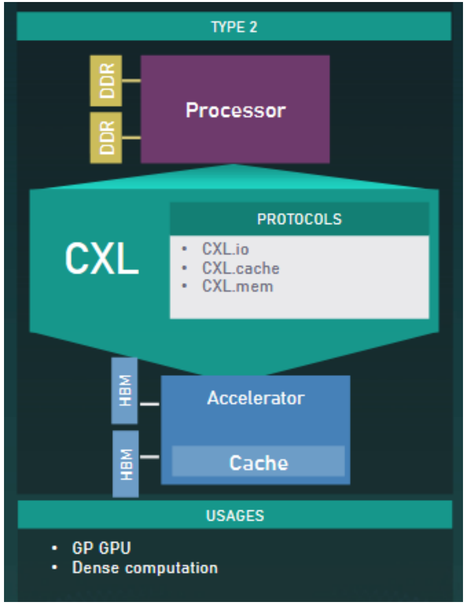
- **Type 1 device**
- Benefits from having a coherency, to perform complex atomic ops
  - e.g., NICs

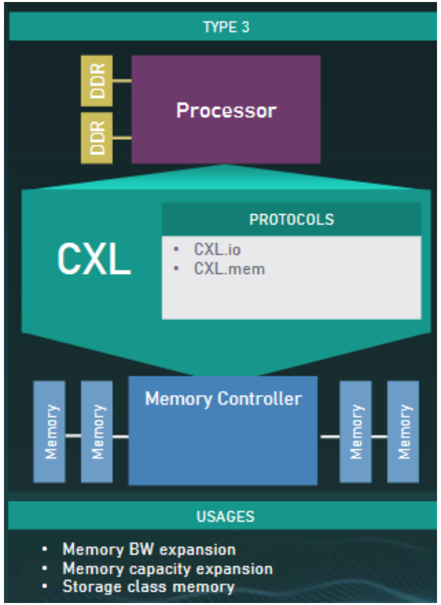
- **Type 2 device**
- Coherency to the host memory
  - e.g., accelerators like GPUs

- **Type 3 device**
- Memory Buffers
  - Memory extensions with
  - different mem media types

https://computeexpresslink.org/wp-content/uploads/2024/03/CXL_3.1-Webinar-Presentation_Feb_2024.pdf

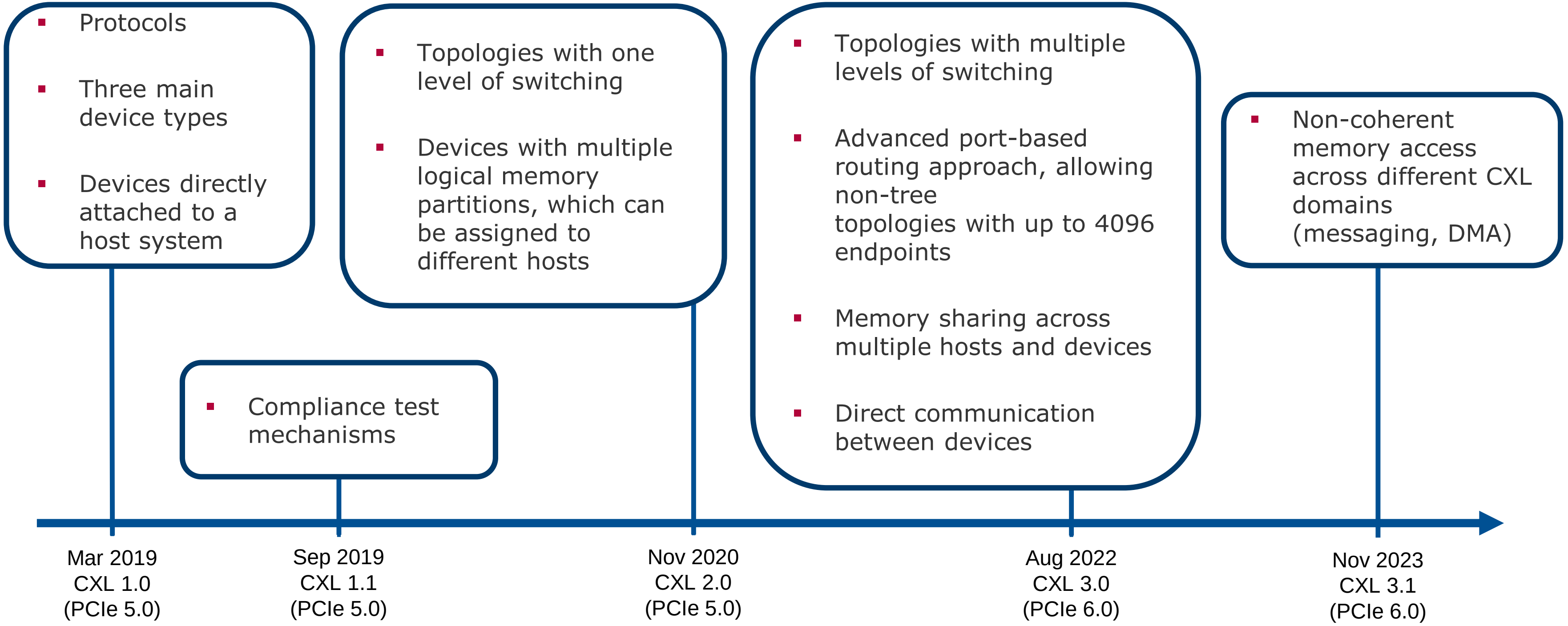






CXL revisions

https://computeexpresslink.org/wp-content/uploads/2024/03/CXL_3.1-Webinar-Presentation_Feb_2024.pdf



- Protocols
- Three main device types
- Devices directly attached to a host system

Mar 2019
CXL 1.0
(PCIe 5.0)

- Compliance test mechanisms

Sep 2019
CXL 1.1
(PCIe 5.0)

- Topologies with one level of switching
- Devices with multiple logical memory partitions, which can be assigned to different hosts

Nov 2020
CXL 2.0
(PCIe 5.0)

- Topologies with multiple levels of switching
- Advanced port-based routing approach, allowing non-tree topologies with up to 4096 endpoints
- Memory sharing across multiple hosts and devices
- Direct communication between devices

Aug 2022
CXL 3.0
(PCIe 6.0)

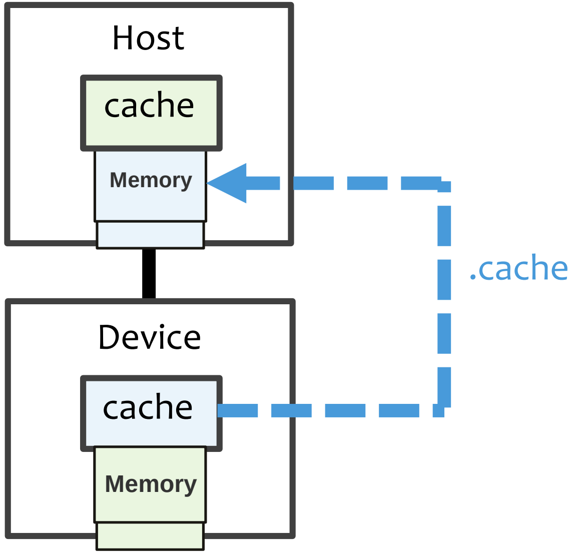
- Non-coherent memory access across different CXL domains (messaging, DMA)

Nov 2023
CXL 3.1
(PCIe 6.0)

Managing Cache Coherence

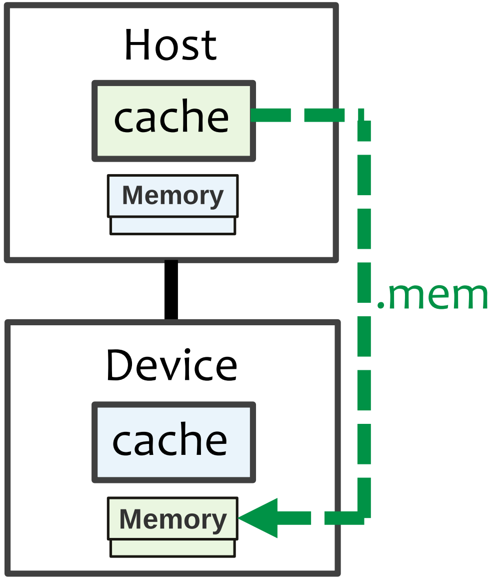
CXL.cache protocol

- Device can cache data stored in host memory
- Using the MESI protocol with a 64 Byte cache line size
- Asymmetric coherence protocol
  - Host manages all tracking of coherence for peer caches
    - Keep the protocol simple at the device
  - Device never directly interacts with any peer cache
  - Device only manages its own cache and sends requests to the host



CXL.mem protocol

- Enables device to expose device memory
- Simple reads and writes from the host to device memory
- The protocol is memory media type independent
  - DRAM, Pmem, HBM, flash memory
- Allows a device to back-invalidate cache line copies with CXL 3.0
- Memory integrated via CXL.mem is part of the unified virtual memory address space.



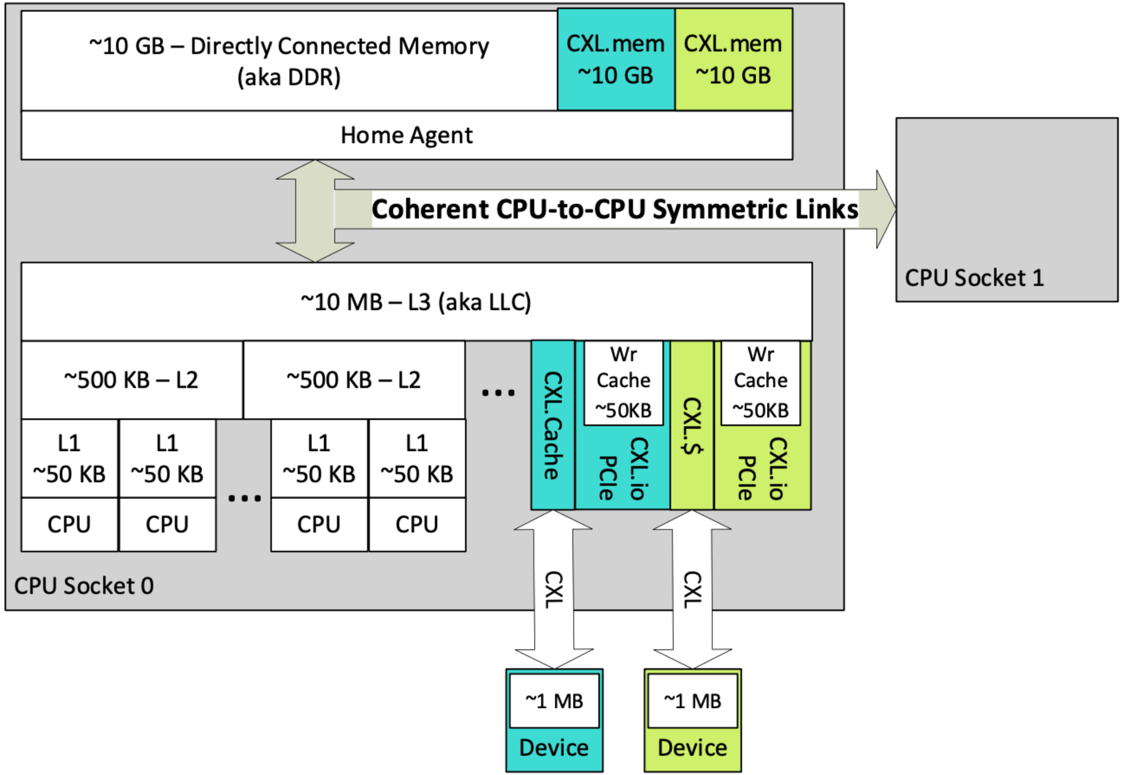
CPU cache hierarchy with CXL

- Multiple levels of coherent caches
- L1: small capacity, lower latency, highest bandwidth

- L2: larger capacity, might be shared between multiple cores

- L3/LLC: higher capacity, higher latency, shared between many CPU cores

- CXL allows devices to directly engage in the cache hierarchy of a CPU below the LLC

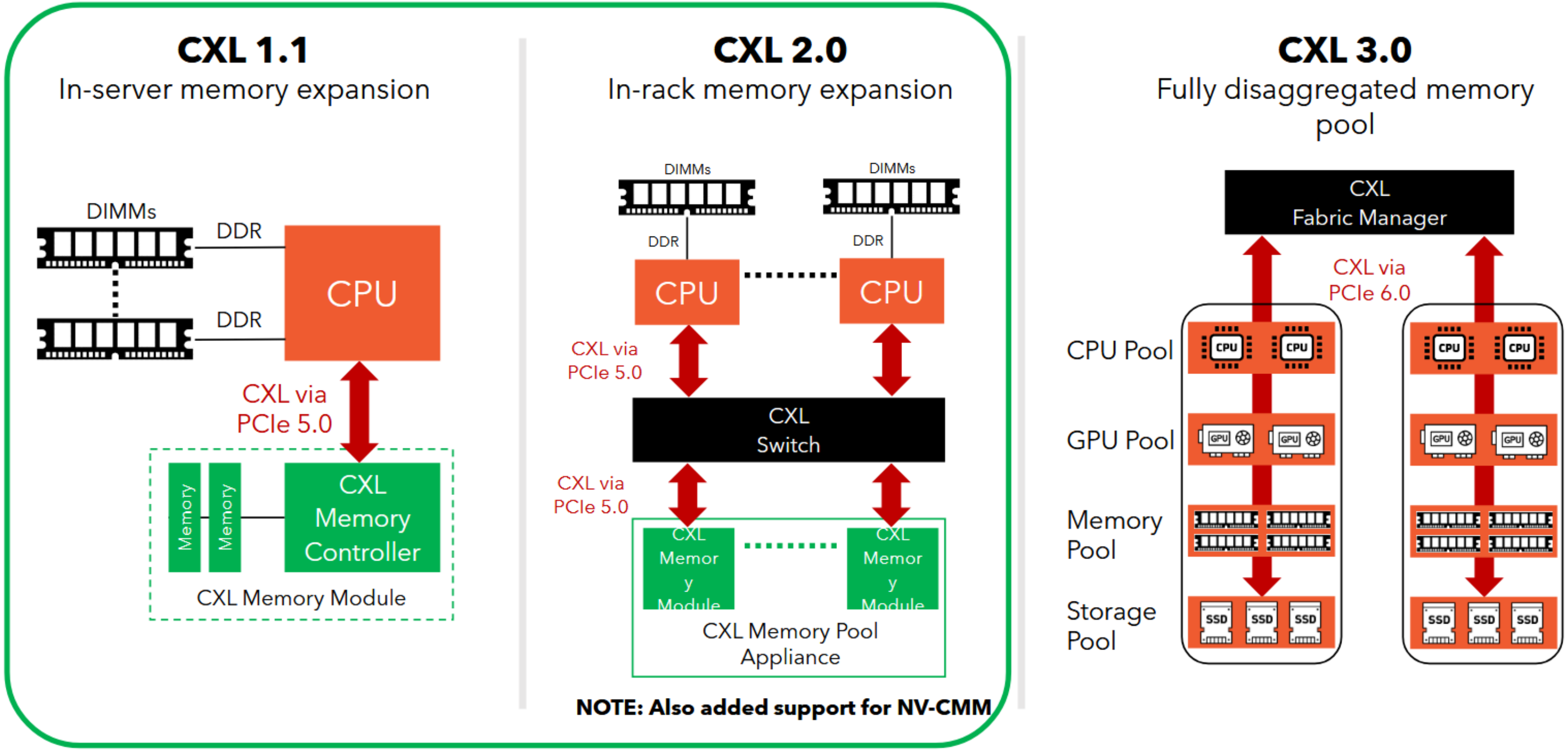


Robert Blankenship. 2020. Compute Express Link (CXL): Memory and Cache Protocols

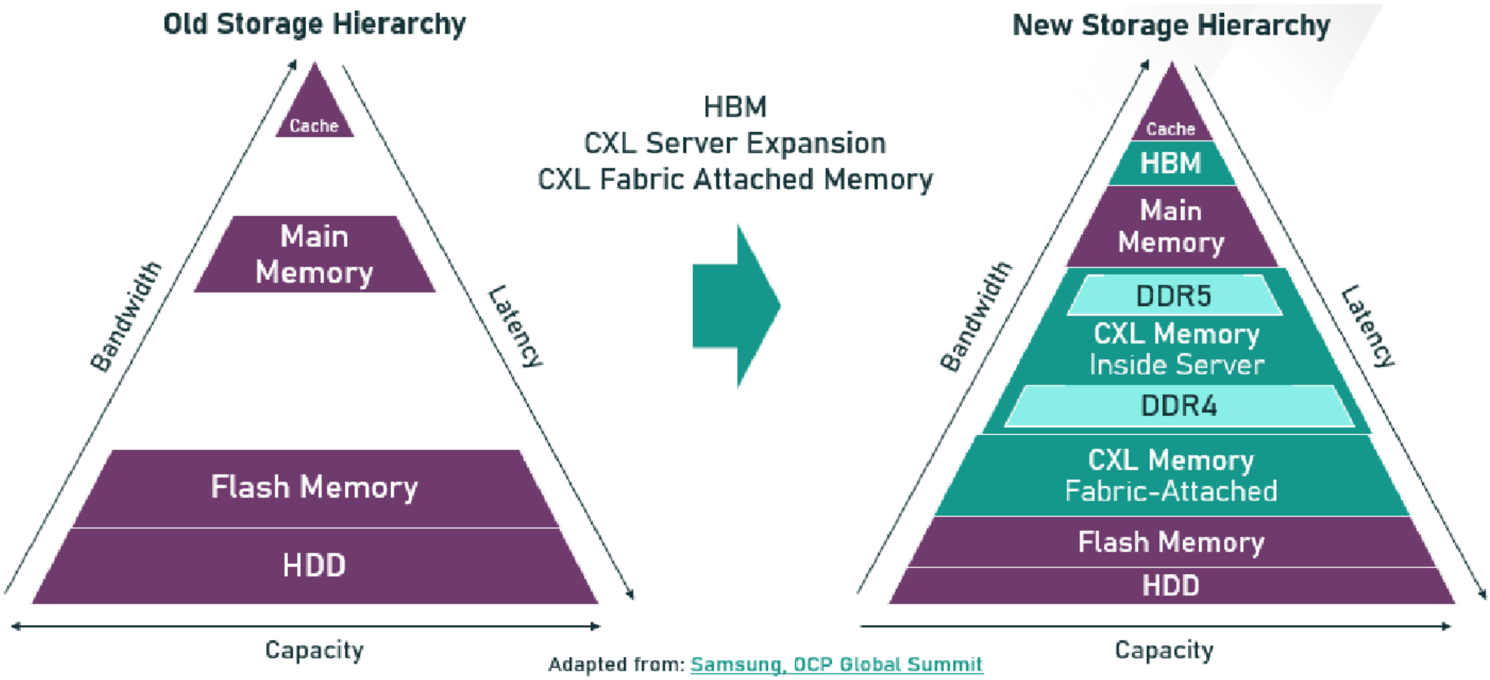
Topologies

Topology Scope

https://snia.org/sites/default/files/CMSC/2025-0326_Unlocking_CXL_Webinar_Final.pdf



Extended memory hierarchy

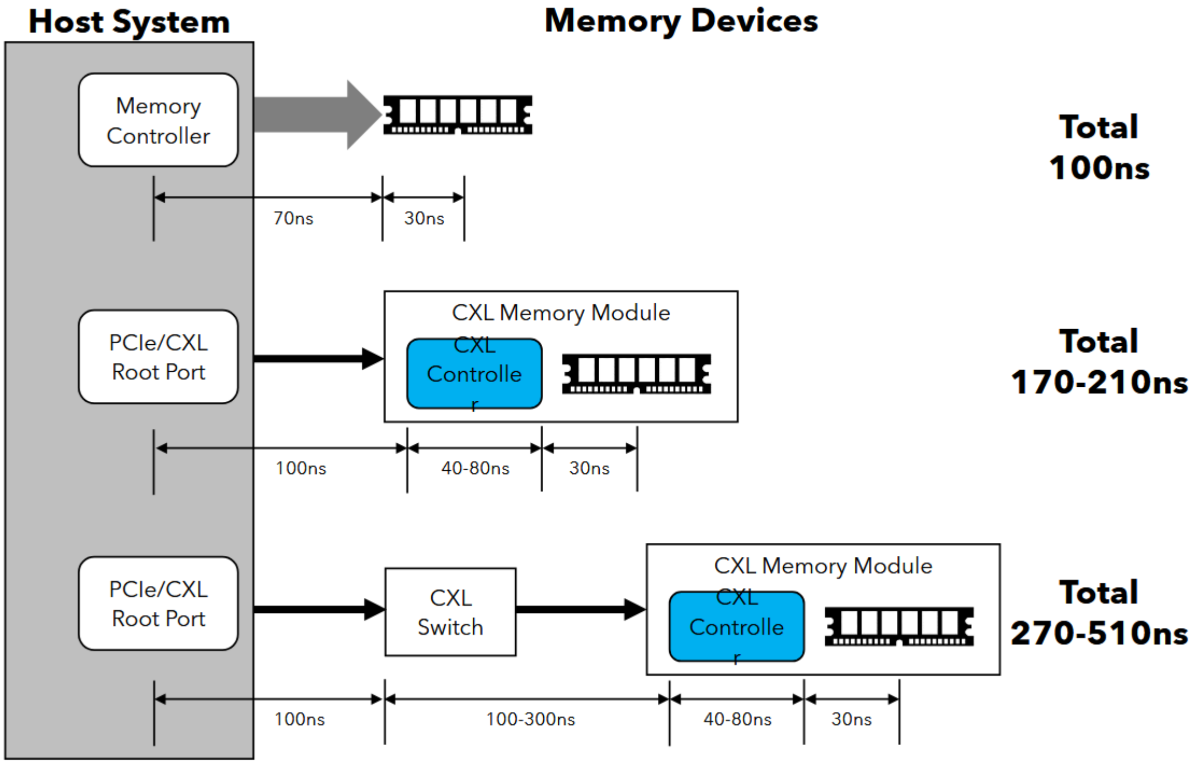


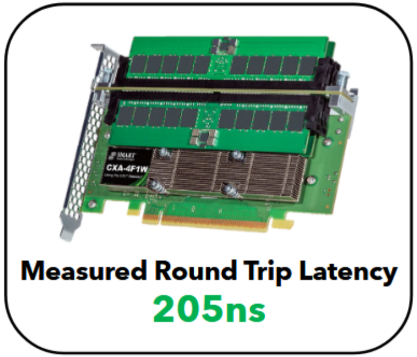
Adapted from: Samsung, OCP Global Summit

https://snia.org/sites/default/files/CMSC/2025-0326_Unlocking_CXL_Webinar_Final.pdf

Demystifying CXL

Performance numbers across modes

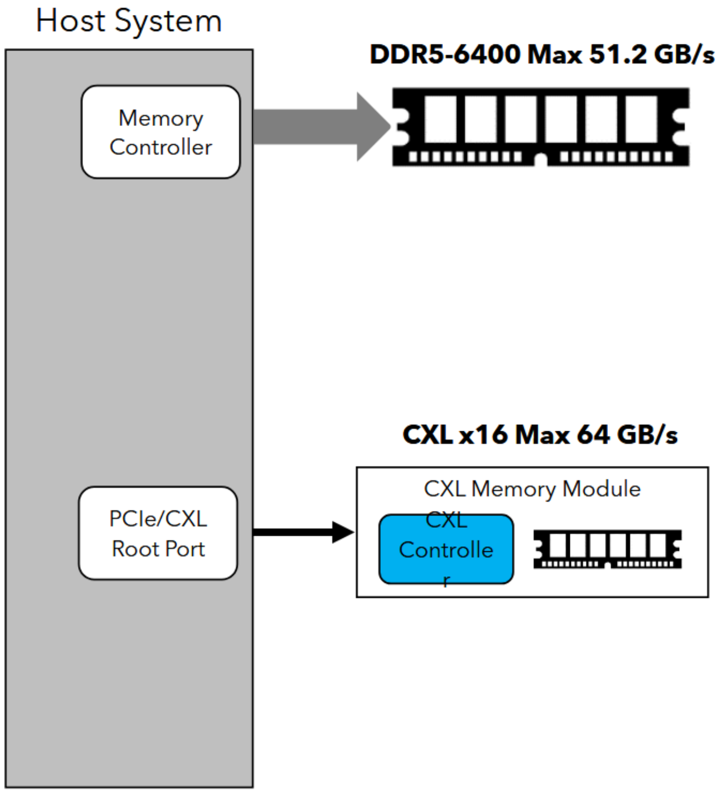


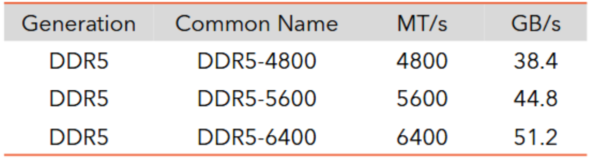


Measured Round Trip Latency
205ns

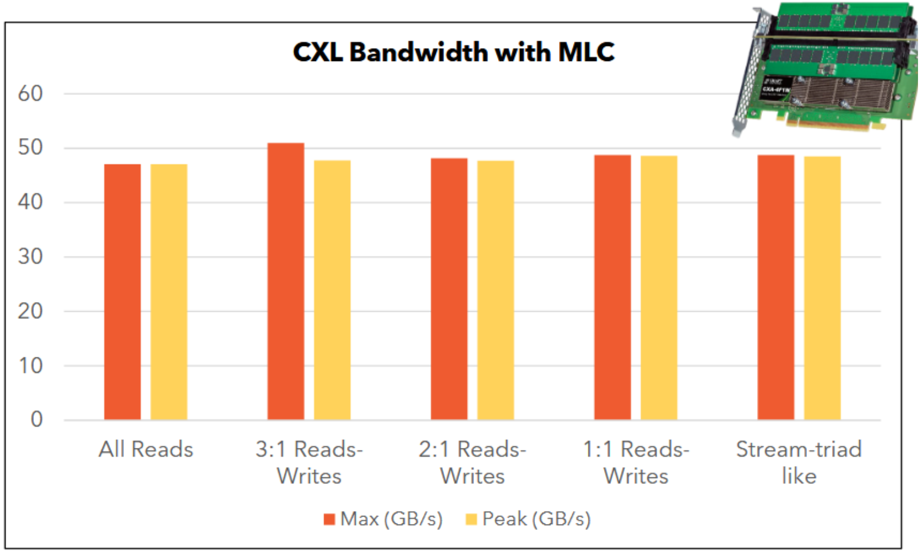
https://snia.org/sites/default/files/CMSC/2025-0326_Unlocking_CXL_Webinar_Final.pdf

Bandwidth across modes





Generation	Common Name	MT/s	GB/s
DDR5	DDR5-4800	4800	38.4
DDR5	DDR5-5600	5600	44.8
DDR5	DDR5-6400	6400	51.2



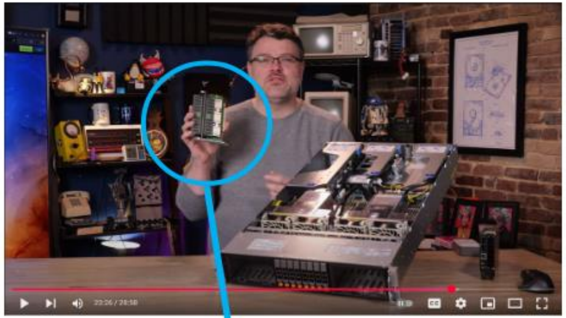
https://snia.org/sites/default/files/CMSC/2025-0326_Unlocking_CXL_Webinar_Final.pdf

Benchmarks (demo)

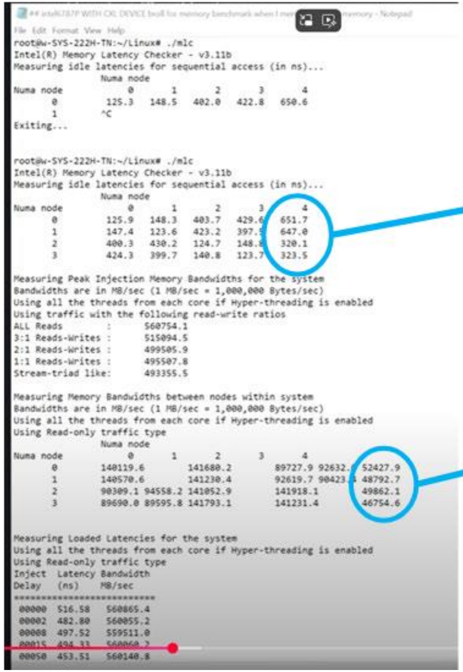
Level1Techs Real World Demo

“Benchmarking the Xeon 6 6787P in the Supermicro 222H-TN”

YouTube



SMART 4-DIMM AIC



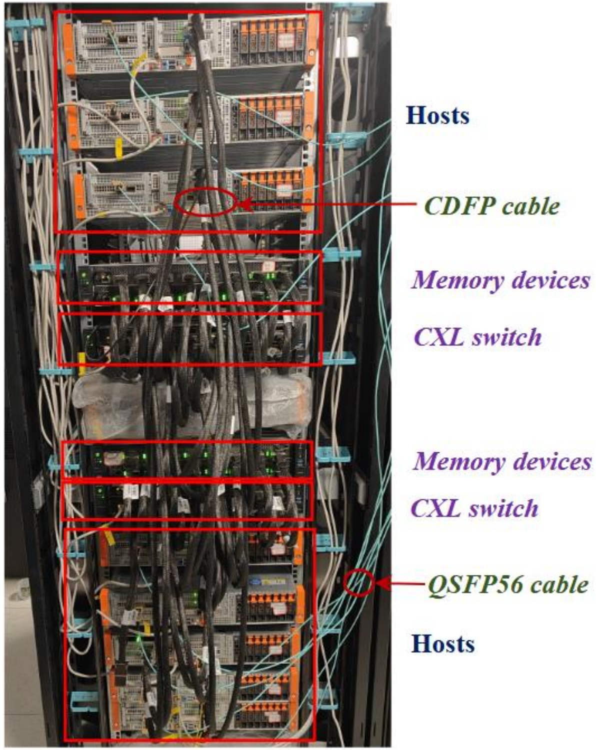
```
root@w-SYS-222H-TN:~/Linux# ./mlc
Intel(R) Memory Latency Checker - v3.11b
Measuring idle latencies for sequential access (in ns)...
                Numa node
Numa node            0       1       2       3       4
       0         125.3   148.5   402.0   422.8   650.6
       1         ^C
Exiting...


root@w-SYS-222H-TN:~/Linux# ./mlc
Intel(R) Memory Latency Checker - v3.11b
Measuring idle latencies for sequential access (in ns)...
                Numa node
Numa node            0       1       2       3       4
       0         125.9   148.3   403.7   429.6   651.7
       1         147.4   123.6   423.2   397.5   647.0
       2         400.3   430.2   124.7   148.8   320.1
       3         424.3   399.7   140.8   123.7   323.5

Measuring Peak Injection Memory Bandwidths for the system
Bandwidths are in MB/sec (1 MB/sec = 1,000,000 Bytes/sec)
Using all the threads from each core if Hyper-threading is enabled
Using traffic with the following read-write ratios
ALL Reads        :      560754.1
3:1 Reads-Writes :      515094.5
2:1 Reads-Writes :      499505.9
1:1 Reads-Writes :      495507.8
Stream-triad like:      493355.5

Measuring Memory Bandwidths between nodes within system
Bandwidths are in MB/sec (1 MB/sec = 1,000,000 Bytes/sec)
Using all the threads from each core if Hyper-threading is enabled
Using Read-only traffic type
                Numa node
Numa node            0       1       2       3       4
       0        140119.6        141680.2        89727.9 92632.0 52427.9
       1        140570.6        141230.4        92619.7 90423.4 48792.7
       2        90309.1 94558.2 141052.9        141918.1        49862.1
       3        89690.0 89595.8 141793.1        141231.4        46754.6


Measuring Loaded Latencies for the system
Using all the threads from each core if Hyper-threading is enabled
Using Read-only traffic type
Inject  Latency Bandwidth
Delay   (ns)    MB/sec
==========================
 00000  516.58   560865.4
 00002  482.80   560055.2
 00008  497.52   559511.0
 00015  494.33   560060.2
 00050  453.51   560140.8
```

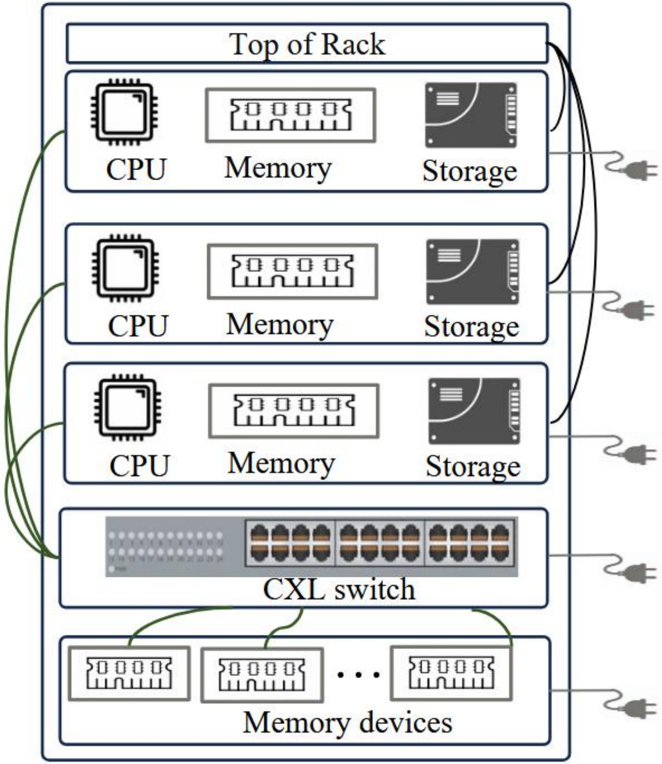
Latency
~320ns

Bandwidth
46 - 52GB/s

https://www.youtube.com/watch?v=8GCd3QSNWLo

First CXL-switch (2.0) deployment





src: Yang et al. Unlocking the Potential of CXL for Disaggregated Memory in Cloud-Native Databases

DRAM vs. CXL vs. RDMA latency comparison

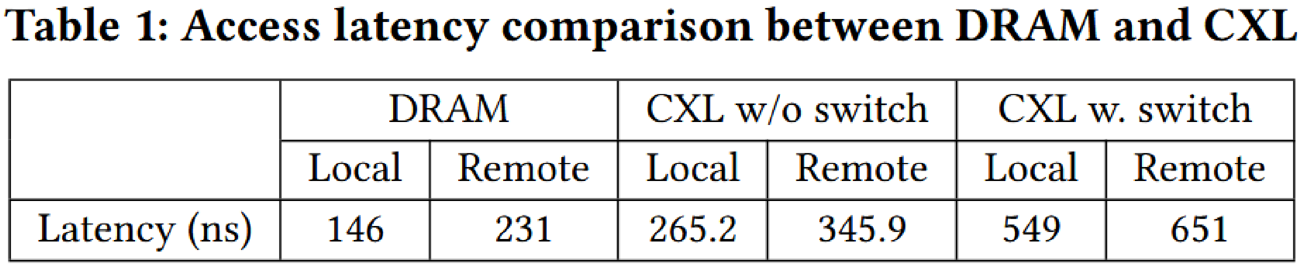


Table 1: Access latency comparison between DRAM and CXL

	DRAM		CXL w/o switch		CXL w. switch	
	Local	Remote	Local	Remote	Local	Remote
Latency (ns)	146	231	265.2	345.9	549	651

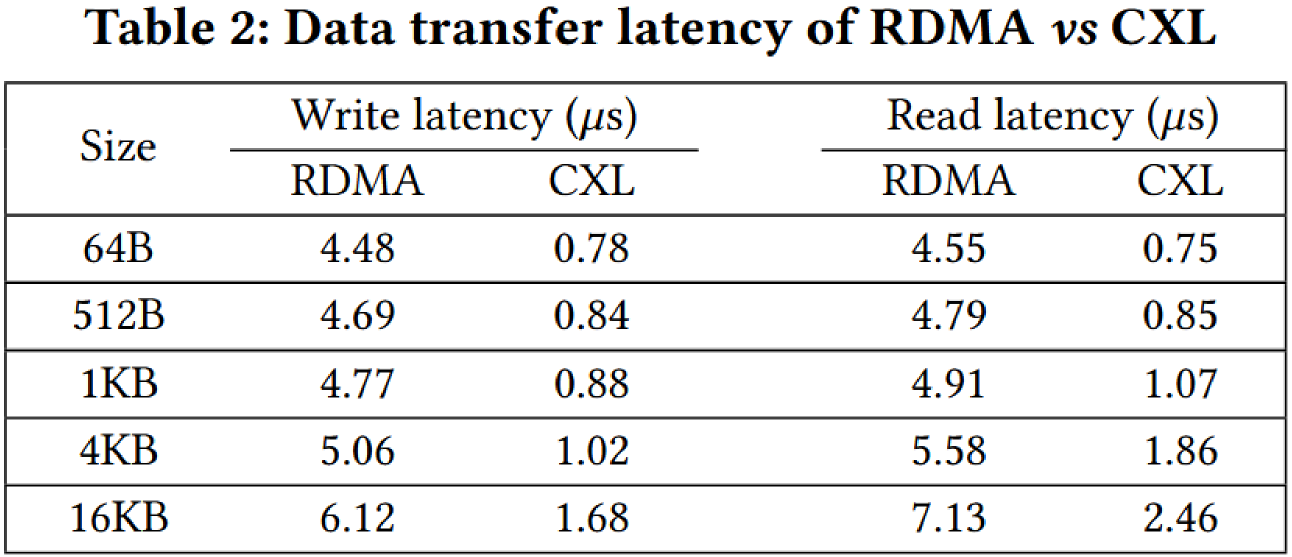


Table 2: Data transfer latency of RDMA *vs* CXL

Size	Write latency (μs)		Read latency (μs)	
	RDMA	CXL	RDMA	CXL
64B	4.48	0.78	4.55	0.75
512B	4.69	0.84	4.79	0.85
1KB	4.77	0.88	4.91	1.07
4KB	5.06	1.02	5.58	1.86
16KB	6.12	1.68	7.13	2.46

src: Yang et al. Unlocking the Potential of CXL for Disaggregated Memory in Cloud-Native Databases

Available hardware

- **Intel**
  - Sapphire Rapids (CXL 1.1) (2023)
  - Emerald Rapids (CXL 1.1) (2023)
  - Granite Rapids (CXL 2.0), released 2024
  - Diamond Rapids (CXl 3.0), expected 2025

- **AMD**
  - AMD Genoa (CXL 1.1) (2022)
  - AMD Bergamo (CXL 1.1) (2023)
  - AMD Turin (CXL 2.0) (2024)
  - AMD Venice (CXL 3.0) (2025)

- **ARM**
  - Neoverse v3: up to 128 crores, CXL 3.0, HBM3

Available Hardware

- **Samsung**
  - CXL memory module – DRAM (CMM-D)
    - x8 PCIe5
    - 256 GB
    - Bandwidth up to 28 GB/s
    - 520 ns average latency
  - CXL memory module – Box (CMM-B)
    - Host up to 24x CMM-D devices
    - Total capacity (3-24 TiB)
    - CXL 1.1 and 2.0 compliant
    - Supports SDL memory pooling

https://semiconductor.samsung.com/news-events/tech-blog/cxl-memory-module-box-cmm-b/

Programming with Device memory (CXL 1.1)

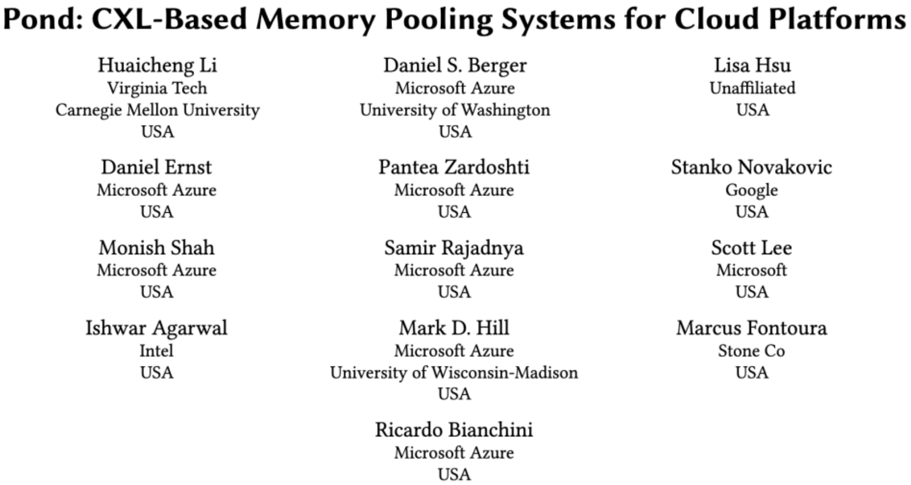
- The OS identifies CXL device memory as memory-only NUMA node

- Allows programmers to utilize NUMA-related system calls to interact with the device memory

- Examples:
  - mbind: set the memory allocation policy for a specific memory regions
  - set_mpolicy: set the memory allocation policy for the calling thread and its children
  - move_pages: move pages of a process to another NUMA node

CXL in research/products

CXL work on cloud platforms



Pond: CXL-Based Memory Pooling Systems for Cloud Platforms

Huaicheng Li
Virginia Tech
Carnegie Mellon University
USA

Daniel S. Berger
Microsoft Azure
University of Washington
USA

Lisa Hsu
Unaffiliated
USA

Daniel Ernst
Microsoft Azure
USA

Pantea Zardoshti
Microsoft Azure
USA

Stanko Novakovic
Google
USA

Monish Shah
Microsoft Azure
USA

Samir Rajadnya
Microsoft Azure
USA

Scott Lee
Microsoft
USA

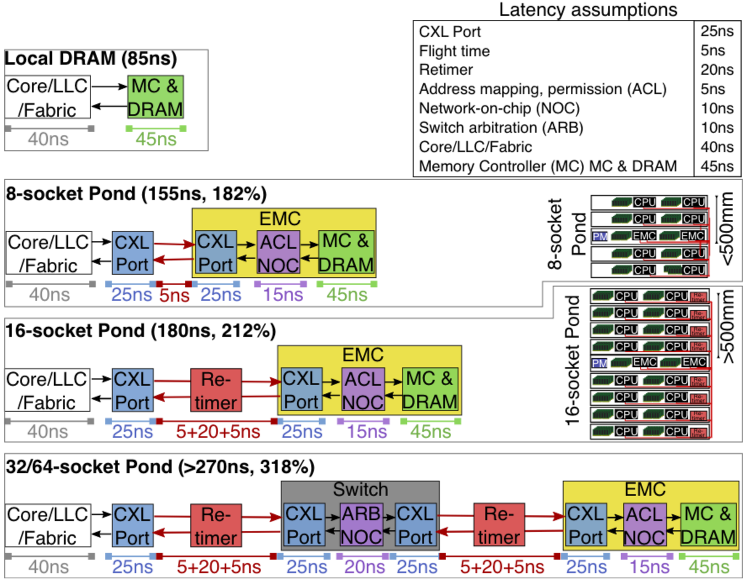
Ishwar Agarwal
Intel
USA

Mark D. Hill
Microsoft Azure
University of Wisconsin-Madison
USA

Marcus Fontoura
Stone Co
USA

Ricardo Bianchini
Microsoft Azure
USA

- ASPLOS 2023
- CXL-based full-stack memory pool for cloud deployment
- Analysis on workload sensitivity to memory latency
- Analysis of the effectiveness and latency of different CXL memory pool sizes
- Prediction model for latency and resource management at datacenter scale
- Evaluation based on emulated CXL memory accessc



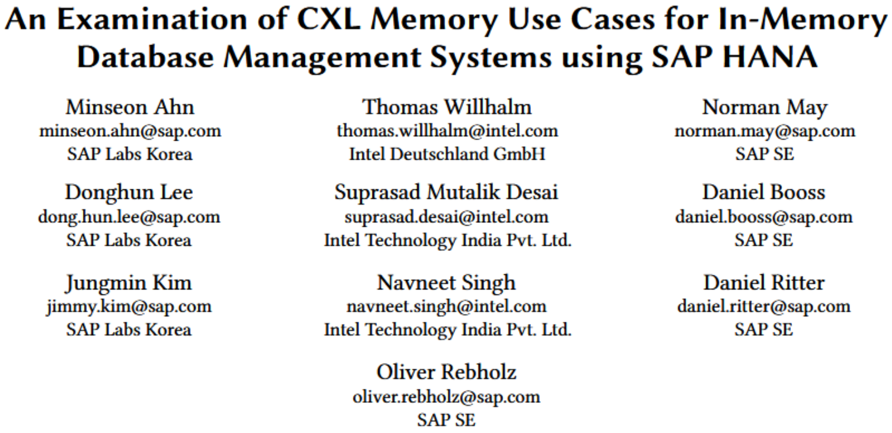
Latency assumptions

CXL Port	25ns
Flight time	5ns
Retimer	20ns
Address mapping, permission (ACL)	5ns
Network-on-chip (NOC)	10ns
Switch arbitration (ARB)	10ns
Core/LLC/Fabric	40ns
Memory Controller (MC) MC & DRAM	45ns

Local DRAM (85ns)
Core/LLC /Fabric
MC & DRAM
40ns
45ns
8-socket Pond (155ns, 182%)
EMC
CXL Port
ACL NOC
25ns 5ns 25ns 15ns 45ns
8-socket Pond
CPU
PM
EMC
<500mm
16-socket Pond (180ns, 212%)
Re-timer
25ns 5+20+5ns 25ns 15ns 45ns
16-socket Pond
>500mm
32/64-socket Pond (>270ns, 318%)
Switch
ARB NOC
25ns 5+20+5ns 25ns 20ns 25ns 5+20+5ns 25ns 15ns 45ns

Figure 7: Pool size and latency tradeoffs (§4.1). Small Pond pools of 8-16 sockets add only 75-90ns relative to NUMA-local DRAM. Latency increases for larger pools that require retimers and a switch.

CXL for in-memory DBMSs using SAP HANA



An Examination of CXL Memory Use Cases for In-Memory Database Management Systems using SAP HANA

Minseon Ahn
minseon.ahn@sap.com
SAP Labs Korea

Thomas Willhalm
thomas.willhalm@intel.com
Intel Deutschland GmbH

Norman May
norman.may@sap.com
SAP SE

Donghun Lee
dong.hun.lee@sap.com
SAP Labs Korea

Suprasad Mutalik Desai
suprasad.desai@intel.com
Intel Technology India Pvt. Ltd.

Daniel Booss
daniel.booss@sap.com
SAP SE

Jungmin Kim
jimmy.kim@sap.com
SAP Labs Korea

Navneet Singh
navneet.singh@intel.com
Intel Technology India Pvt. Ltd.

Daniel Ritter
daniel.ritter@sap.com
SAP SE

Oliver Rebholz
oliver.rebholz@sap.com
SAP SE

- VLDB 2024
- Dynamic memory expansion with CXL memory devices for in-memory RDBMSs
- Performance impact of increased latency/lower bandwidth depends on memory access patterns of data structures
- Feasibility of CXL shared memory to improve restart times during failover.
- CXL shows almost no performance degradation for OLTP and 40-84% reduction of restart times.

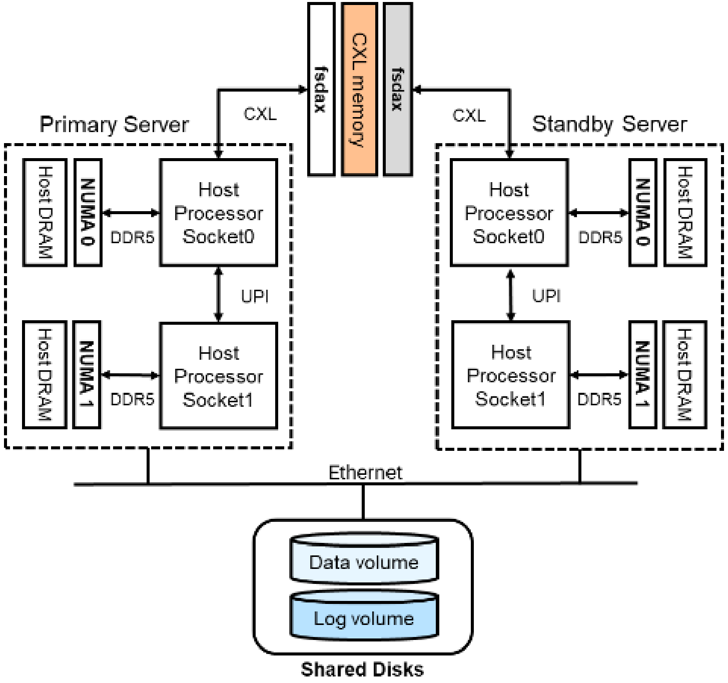
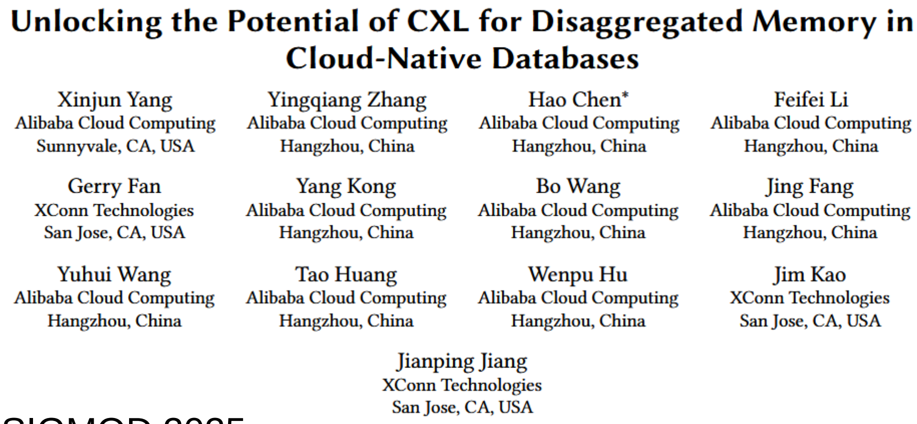


Figure 10: System overview for failover with fast restart

CXL for disaggregated Cloud-Native DBs



Unlocking the Potential of CXL for Disaggregated Memory in Cloud-Native Databases

Xinjun Yang
Alibaba Cloud Computing
Sunnyvale, CA, USA

Yingqiang Zhang
Alibaba Cloud Computing
Hangzhou, China

Hao Chen*
Alibaba Cloud Computing
Hangzhou, China

Feifei Li
Alibaba Cloud Computing
Hangzhou, China

Gerry Fan
XConn Technologies
San Jose, CA, USA

Yang Kong
Alibaba Cloud Computing
Hangzhou, China

Bo Wang
Alibaba Cloud Computing
Hangzhou, China

Jing Fang
Alibaba Cloud Computing
Hangzhou, China

Yuhui Wang
Alibaba Cloud Computing
Hangzhou, China

Tao Huang
Alibaba Cloud Computing
Hangzhou, China

Wenpu Hu
Alibaba Cloud Computing
Hangzhou, China

Jim Kao
XConn Technologies
San Jose, CA, USA

Jianping Jiang
XConn Technologies
San Jose, CA, USA

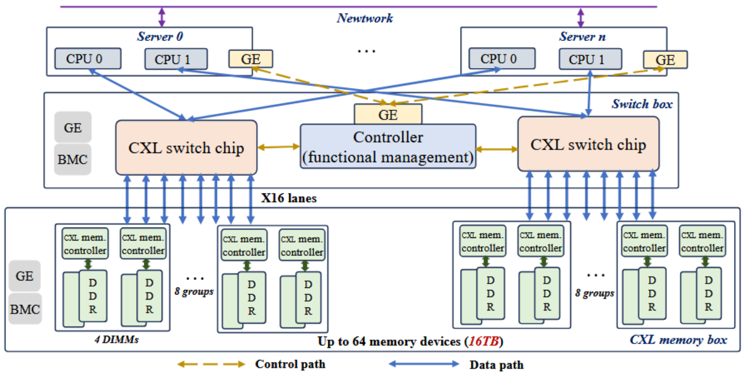


Figure 5: Physical topology of *PolarCXLMem* deployment.

- SIGMOD 2025
- First commercial deployment/academic report with CXL 2.0 switch-based disaggregated memory system
- Supports novel instant recovery scheme, fast buffer pool warm-up after a crash.
- Presents a new software-based cache-coherency protocol to facilitate data sharing between multi-primary database nodes
- Can improve throughput up to 2x in pooling, and 1.55x in sharing scenarios compared to RDMA-based solutions

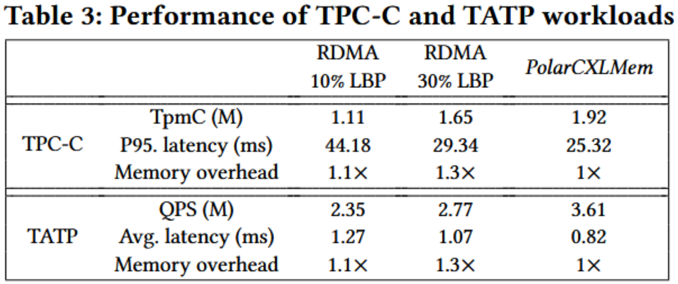
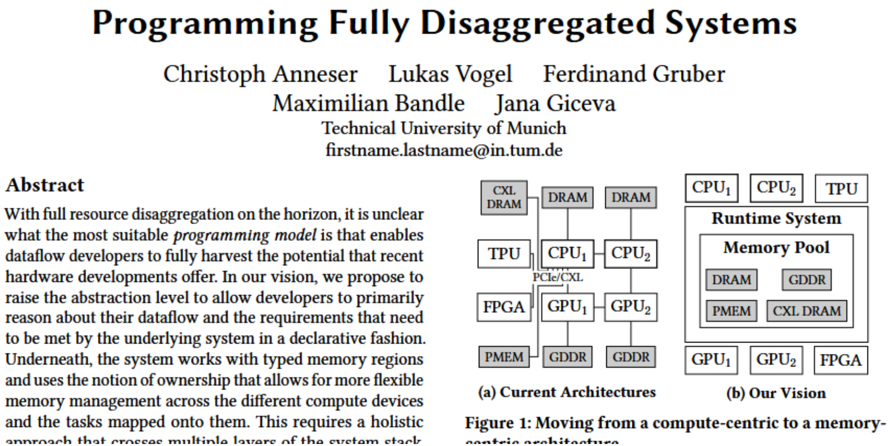


Table 3: Performance of TPC-C and TATP workloads

		RDMA 10% LBP	RDMA 30% LBP	*PolarCXLMem*
TPC-C	TpmC (M)	1.11	1.65	1.92
	P95. latency (ms)	44.18	29.34	25.32
	Memory overhead	1.1×	1.3×	1×
TATP	QPS (M)	2.35	2.77	3.61
	Avg. latency (ms)	1.27	1.07	0.82
	Memory overhead	1.1×	1.3×	1×

Towards memory-centric DBs and programming model

- From our research group:



Programming Fully Disaggregated Systems

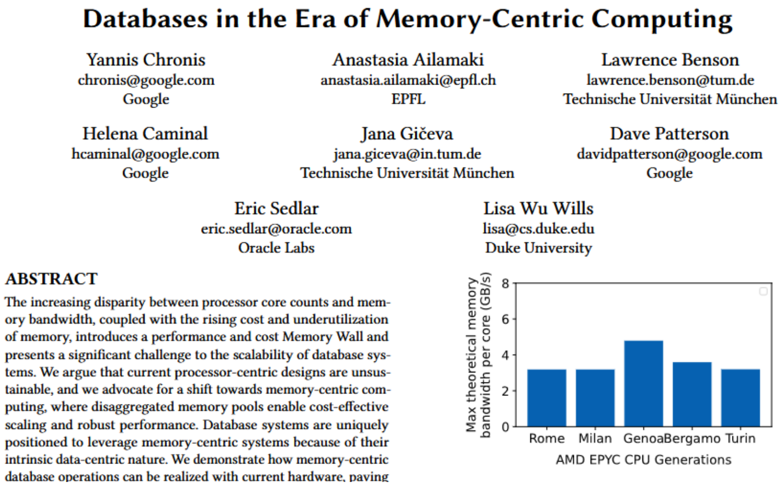
Christoph Anneser Lukas Vogel Ferdinand Gruber
Maximilian Bandle Jana Giceva
Technical University of Munich
firstname.lastname@in.tum.de

Abstract

With full resource disaggregation on the horizon, it is unclear what the most suitable *programming model* is that enables dataflow developers to fully harvest the potential that recent hardware developments offer. In our vision, we propose to raise the abstraction level to allow developers to primarily reason about their dataflow and the requirements that need to be met by the underlying system in a declarative fashion. Underneath, the system works with typed memory regions and uses the notion of ownership that allows for more flexible memory management across the different compute devices and the tasks mapped onto them. This requires a holistic approach that crosses multiple layers of the system stack

(a) Current Architectures (b) Our Vision

Figure 1: Moving from a compute-centric to a memory-centric architecture



Databases in the Era of Memory-Centric Computing

Yannis Chronis
chronis@google.com
Google

Anastasia Ailamaki
anastasia.ailamaki@epfl.ch
EPFL

Lawrence Benson
lawrence.benson@tum.de
Technische Universität München

Helena Caminal
hcaminal@google.com
Google

Jana Gičeva
jana.giceva@in.tum.de
Technische Universität München

Dave Patterson
davidpatterson@google.com
Google

Eric Sedlar
eric.sedlar@oracle.com
Oracle Labs

Lisa Wu Wills
lisa@cs.duke.edu
Duke University

ABSTRACT

The increasing disparity between processor core counts and memory bandwidth, coupled with the rising cost and underutilization of memory, introduces a performance and cost Memory Wall and presents a significant challenge to the scalability of database systems. We argue that current processor-centric designs are unsustainable, and we advocate for a shift towards memory-centric computing, where disaggregated memory pools enable cost-effective scaling and robust performance. Database systems are uniquely positioned to leverage memory-centric systems because of their intrinsic data-centric nature. We demonstrate how memory-centric database operations can be realized with current hardware, paving

References

- Lecture notes from *Hardware-conscious Data Processing (HPI) – Marcel Weisgut*

- CXL consortium
  - An Introduction to Computer Express Link (CXL) Technology (link)
  - Introducing the CXL 3.1 Specification (link)
  - Opportunities and Challenges for CXL (link)
- SNIA consortium
  - Unlocking CXL's Potential: Revolutionizing Server Memory and Performance (link)

- *Research papers:*
  - *Li et al.* "Pond: CXL-Based Memory Pooling Systems for Cloud Platforms"
  - *Ahn et al.* "An Examination of CXL use-cases for INDBMS using SAP HANA" VLDB 2024
  - *Yang et al.* "Unlocking the Potential of CXL for Disaggregated Memory in Cloud-Native Databases SIGMOD 2025 (Best paper award for industry track)
  - Zhong et al. "My CXL Pool Obviates Your PCIe Switch" HotOS 2025