Increasing AI and HPC Application Performance with CXL Fabrics

Presented by

Kurtis Bowman, AMD

Sandeep Dattaprasad, Astera Labs

Steve Scargall, MemVerge

SNIA COMPUTE, MEMORY, AND STORAGE SUMMIT

Solutions, Architectures, and Community

VIRTUAL EVENT, MAY 21-22, 2024

Meet your panelists

Kurtis Bowman
CXL MWG Co-Chair
Director, Server System Performance at AMD

Sandeep Dattaprasad
Senior Product Manager at Astera Labs

Steve Scargall
Senior Product Manager & Software Architect
at MemVerge

CXL Ecosystem

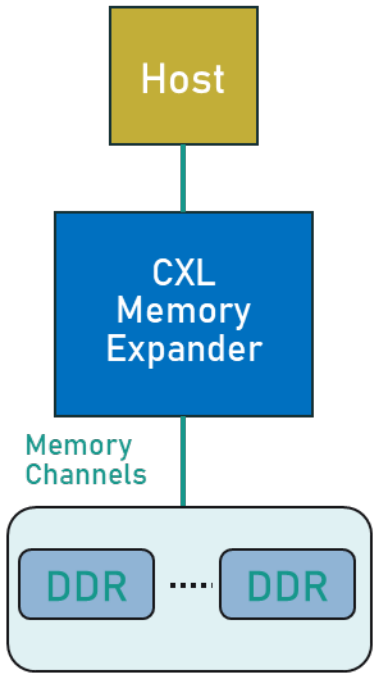
Growth of CXL ecosystem since its inception

- Ecosystem that meets the ever-increasing performance and scale requirements
- Fully backwards compatible
- Lower overall system cost
- Comprehensive compliance and testing support
- Products being deployed

CXL Specification Feature Summary

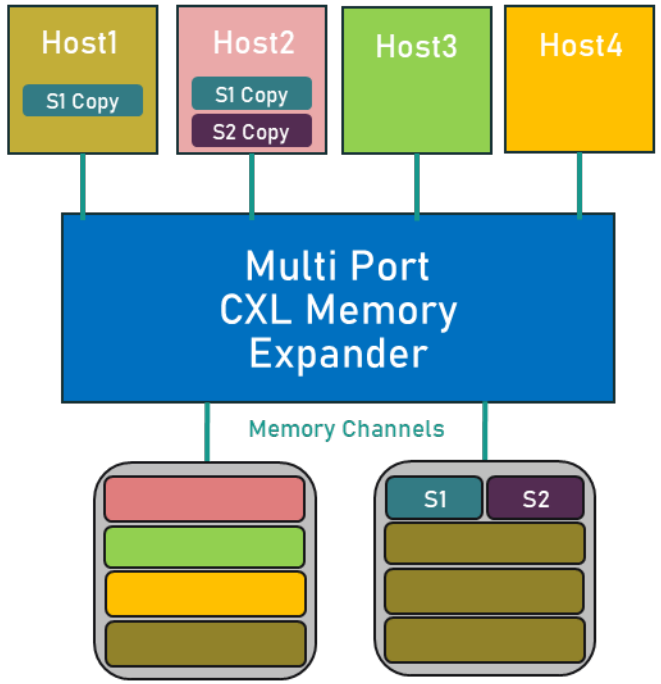
Features	CXL 1.0 / 1.1	CXL 2.0	CXL 3.0	CXL 3.1
Release date	2019	2020	August 2022	November 2023
Max link rate	32GTs	32GTs	64GTs	64GTs
Flit 68 byte (up to 32 GTs)	✓	✓	✓	✓
Flit 256 byte (up to 64 GTs)			✓	✓
Type 1, Type 2 and Type 3 Devices	✓	✓	✓	✓
Memory Pooling w/ MLDs		✓	✓	✓
Global Persistent Flush		✓	✓	✓
CXL IDE		✓	✓	✓
Switching (Single-level)		✓	✓	✓
Switching (Multi-level)			✓	✓
Direct memory access for peer-to-peer			✓	✓
Enhanced coherency (256 byte flit)			✓	✓
Memory sharing (256 byte flit)			✓	✓
Multiple Type 1/Type 2 devices per root port			✓	✓
Fabric capabilities (256 byte flit)			✓	✓
Fabric Manager API definition for PBR Switch				✓
Host-to-Host communication with Global Integrated Memory (GIM) concept				✓
Trusted-Execution-Environment (TEE) Security Protocol				✓
Memory expander enhancements (up to 32-bit of meta data, RAS capability enhancements)				✓

CXL 3.1 Use Cases for AI & HPC



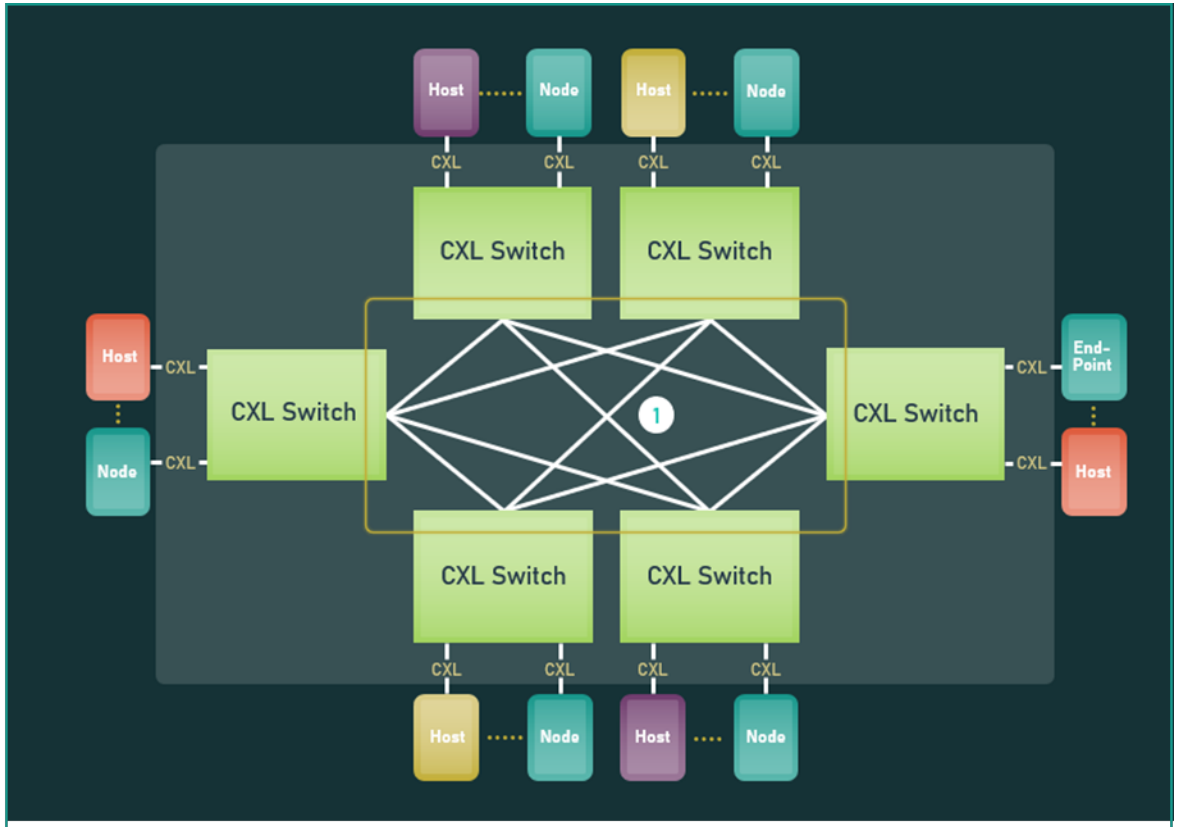
CXL Memory Expansion

- Increases Capacity for AI applications
- Improves processor efficiency
- Improves Bandwidth for GPUs
- Lowers TCO- higher memory capacity for cheaper $/GB



Pooled Memory *Shared Memory*

- Increases Capacity for AI applications
- Reduces Memory Stranding
- Improves Data Flow Efficiency
- Improves Memory Utilization
- Lowers TCO



- Flexibility across the fabric for HPC use cases
- Scales to large datasets
- Improves Memory Utilization
- Lowers TCO

CXL 3.1 Trusted Security Protocol (TSP)

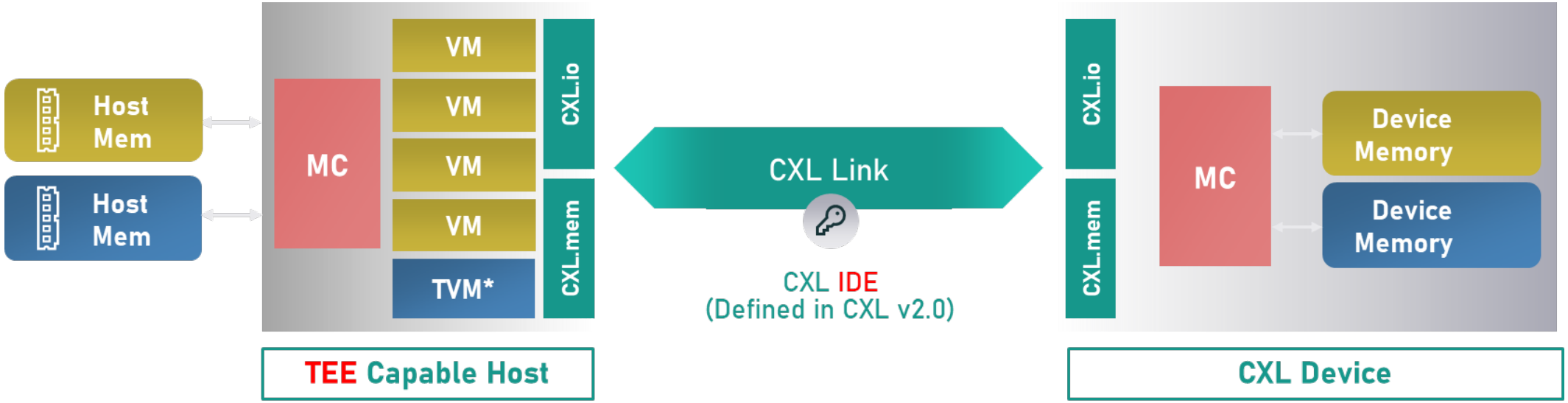
Allows for Virtualization-based, Trusted Execution Environments (TEEs)
to host Confidential Computing Workloads (CC WL)

Key Capabilities:

- Separation between TVM* & CSP's infrastructure (VMM)
- Configuration of CXL device
- Encryption of sensitive data in both Host/Device memory
- Cryptographically verify correct configuration of trusted computing environment

Benefits:

- Freedom to migrate sensitive WLs to TSP-enabled Clouds
- Collaboration with multiple parties for sharing data
- Conform to Compliance & Data sovereignty programs
- Strengthen Application security & Software IP protection



*TVM = Trusted VM

Please take a moment to rate this session.

Your feedback is important to us.

Backup

Igniting Innovation: The Expanding CXL Software Universe

Recent Highlights:

- Linux Kernel and QEMU continue to add CXL features - up to 3.1
  - QEMU has CXL expansion and sharing. DCD and MHD are in development.
  - Kernel 6.8 added NUMA QoS: read/write latency and bandwidth for devices
  - Kernel 6.9 added Weighted NUMA Interleaving
- Fabric Management:
  - Jack Rabbit Labs open-source Fabric Management, Orchestration, and CXL Switch Emulation (CSE)
  - Liqid Composable Infrastructure
  - RedFish
  - And more…
- FAMFS is a scale-out shared memory file system for CXL 3.x
- MemVerge Shared Memory Object Store (GISMO)
- Memory Tiering
  - Kernel TPP, Kernel Weighted NUMA Interleaving, MemVerge Memory Machine (Latency & Bandwidth), Platform BIOS options, DAMON, …

CXL.mem for AI/ML Workloads

- **CXL Memory Expansion:** Scale-up enables larger models and datasets to be processed more efficiently.
- **Efficient GPU Offloading Strategies:** Efficiently offload compute-heavy AI/ML tasks to GPUs, while minimizing data transfer overheads between CPUs and GPUs. E.g.: KV Caches, …
- **Shared Virtual Memory Systems:** Allowing GPUs and CPUs to access a unified memory address space can reduce data movement overhead.
- **Dynamic Data Placement:** Based on access patterns and workload requirements, dynamically place data in the most appropriate memory type (GPU local memory, DRAM, CXL memory).
- **Improve GPU Utilization:** Time Slicing or GPU Fractioning allows multiple users to share the GPU resource. Data is context-switched between the GPU and main memory.
- **Near-Data Processing:** Use CXL Pooling and DCD to move the data to compute using intelligent schedulers and orchestrators.

