CXL Memory Expansion: A Closer Look on Actual Platform

Micron
Vinicius Petrucci, Eishan Mirakhur, Nikesh Agarwal, Su Wei Lim, Vishal Tanna

AMD
Rita Gupta, Mahesh Wagh

Motivation for CXL Memory

The proliferation of data-intensive workloads has led to a significant increase in the volume of data that computing systems need to handle. This expanding data landscape necessitates memory solutions with larger capacities and higher bandwidths. Nevertheless, numerous challenges must be overcome to ensure that the current systems can meet the growing demands for application performance.

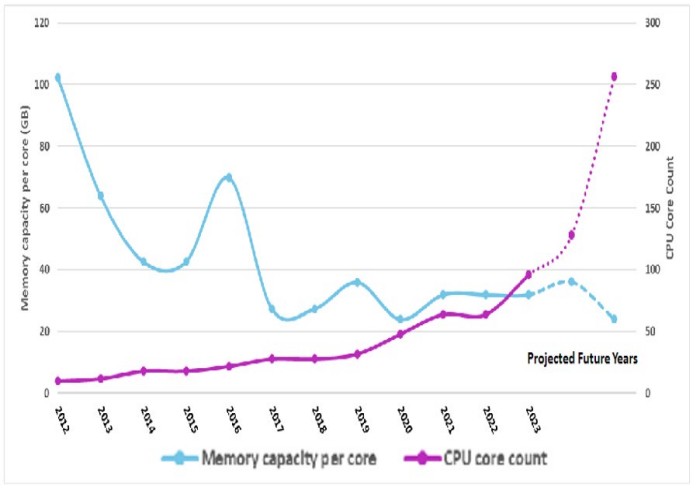
Challenge 1: Hitting the Memory Wall

The memory wall problem continues to pose a fundamental challenge to system performance, as the scaling of CPU cores has outpaced the scaling of memory capacity and bandwidth per core over recent years. If these trends persist, workloads that heavily rely on memory will encounter a bottleneck caused by the movement of data, effectively hitting a performance barrier. CPU vendors have tried addressing this by increasing the number of memory channels per socket. However, the addition of more memory channels comes at a cost of significant real estate in terms of pins and routing complexity, making it challenging to continue increasing the number of channels to keep up with higher and higher core counts. Furthermore, DRAM manufacturers are also facing challenges to scale the density on the memory chips in a pace that matches the core count increase. To increase capacity through alternate means, 2DPC (2-DIMMs per channel) has been explored but it comes at the expense of reduced bandwidth due to signal integrity issues. Other types of memory packaging technologies, such as 3D stacking, are not cost-effective. Hence, memory has increasingly become an extremely critical bottleneck and valuable computing resource within the system.

Year	CPU core count	Theoretical DDR memory data rate (GB/s)	Memory channels	System memory bandwidth (GB/s)	Memory bandwidth per core (GB/s)
2010	8	10.7	2	21.3	2.7
2011 2012	10	10.7	4	42.7	4.3
2013	12	14.9	4	59.7	5.0
2014	18	14.9	4	59.7	3.3
2015	18	17.1	4	68.3	3.8
2016	22	19.2	4	76.8	3.5
2017 2018	28	21.3	6	128.0	4.6
2019	32	23.5	6	140.8	4.4
2020	48	25.6	6	153.6	3.2
2021 2022	64	25.6	8	204.8	3.2
2023	96	38.4	12	460.8	4.8
Future	256*	70.4*	12	844.8	3.3

(CPU vendors added more memory channels in years 2011, 2017, 2021 and 2023)
(Projected increases in core count and memory speeds from public statements and JEDEC specifications)*

(a)



(b)

Fig. 1. a) Progression of CPU and memory and effect on bandwidth/core. b) Historical trends for memory capacity and CPU core count.

Challenge 2: Latency and Capacity Gap in Memory Hierarchy

A significant performance gap has been shaped between the main memory and storage media. Huge latency gap exists in the memory hierarchy between dynamic random-access memory (DRAM) and storage media such as solid-state drive (SSD). With the growing demand for data, complex workloads expect more memory capacity to fit the data sets. Since the industry has not been able to scale memory capacity in a cost-effective way with demand due to technological challenges, data-intensive workloads dealing with large data sets may experience excessive demand paging and IO activities, which can translate to significantly higher data access latency and lower performance**.**

Challenge 3: Increased TCO for Data Centers

The growing significance of memory as a bottleneck has resulted in increased total cost of ownership (TCO) for servers. In fact, according to recent studies [1] conducted on a large-scale cloud computing system, memory accounts for approximately 50% of the overall server cost. As mentioned earlier, aside from the per-socket limitations, the current methods to enhance bandwidth involve raising the data rate, while scaling up capacity is achieved through techniques like

2DPC or increasing DRAM density, each of which are coming at the expense of the other. An alternative approach is to scale out by adding more CPUs or through 3D stacking memory packaging technology, but these approaches come at the expense of much higher TCO.

Compute Express Link™ (CXL) Memory: Attractive Alternative for Scale-Up Expansion

To help alleviate the above-mentioned challenges, the industry is rapidly converging towards a new and fast coherent interconnect protocol, Compute Express Link™ (CXL). CXL is an open industry standard interconnect offering coherency and memory semantics using high bandwidth, low latency connectivity between processors, accelerators, memory, storage, and other IO devices. It helps address the above challenges by scaling up through increased bandwidth and capacity per core in the system. CXL is gaining faster adoption because it introduces load/store semantics to PCI Express (PCIe) physical layer, enabling expansion of both capacity and bandwidth at access latency comparable to remote non uniform memory access (NUMA) node DRAM memory. This addresses the processor pin count challenge and avoids the TCO associated with adding new processors just to get additional bandwidth and capacity. With the introduction of CXL Type-3 memory expansion (CXL.mem), new memory models can be adopted to support increased memory capacity requirements. It also reduces data center TCO for system configurations with similar performance targets.

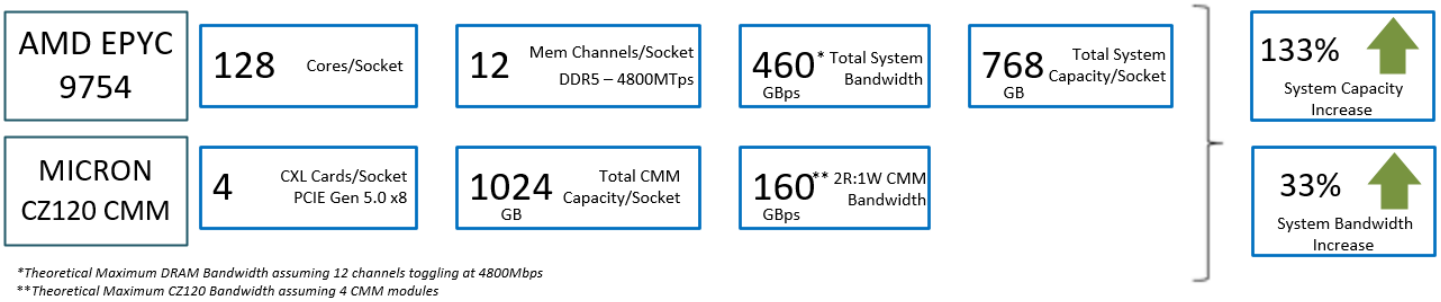
Two important value propositions that CXL-attached memory device brings to the table follow:

- **CXL–Memory Bandwidth Expansion** via heterogeneous interleaving (incarnated at software level).
- **CXL–Memory Capacity Expansion** through memory tiering, commonly leveraging OS level support built upon the well-researched NUMA abstractions and interfaces.

The intent of this paper is to showcase how CXL can help overcome existing systems challenges by adding memory bandwidth and capacity beyond server DIMM slots. We examine the value propositions enabled by CXL from the perspective of experimental data collected executing data-intensive and high-performance workloads on actual server platform supporting multiple CXL memory modules (CMM). This contrasts with prior CXL Memory Expansion experiments mostly executed on early-stage prototypes or emulation/simulation frameworks, which do not necessarily capture the real behavior of real CXL devices running actual workloads in a real system. Running the workload on a real platform and real CMM is an important milestone towards demonstrating the value proposition of CXL in data centers.

System Details and Configurations

Fig. 2 provides the details on Micron™ CMM enabled by the cloud-oriented AMD EPYC™ 9754 CPU that offers up to 128 cores for compute-intensive cloud servers. AMD EPYC 9754 is optimized around higher throughput workloads. It supports up to 12 memory channels allowing a total of 768GB with 64GB memory module per channel or 1152GB capacity with 96GB memory module per channel. With increased core count on EPYC 9754, capacity and bandwidth per core is limited to 6GB/core (64GB memory module per channel) or 9GB/core (96GB memory module per channel) and 3.6 GBps/core assuming 4800 MT/s. With the addition of Micron's 4 x CZ120 CXL Memory Modules with 256GB capacity each, the system capacity and bandwidth per-core increases to 14GB/core (64GB memory module) or 17GB/core (96GB memory module) and 4.85 GBps/core, respectively. Specifically, for the system with 64GB memory module, this is an increase of 133% in memory capacity relative to native DRAM setup (64GB DDR5 per channel) and 33% in bandwidth expansion compared to the system without CXL. While system developer is free to select any local DRAM module capacity available, one possible configuration is highlighted in Fig2 below.



*Theoretical Maximum DRAM Bandwidth assuming 12 channels toggling at 4800Mbps
**Theoretical Maximum CZ120 Bandwidth assuming 4 CMM modules

Fig. 2. AMD platform and Micron CZ120 CMM high-level product details.

The CZ120 CXL Memory Module (CMM) is the first generation of Micron's CXL Memory Expansion product. Each CMM exposes a PCIE Gen5 x8 link width. It has two DDR4 memory channels connected as its backend media, offers a capacity of up to 256GB per CMM and targets a competitive bandwidth for 2-Read:1-Write traffic pattern with a low pin-to-pin unloaded latency. Fig. 3 highlights the key differentiation features for CZ120 product.

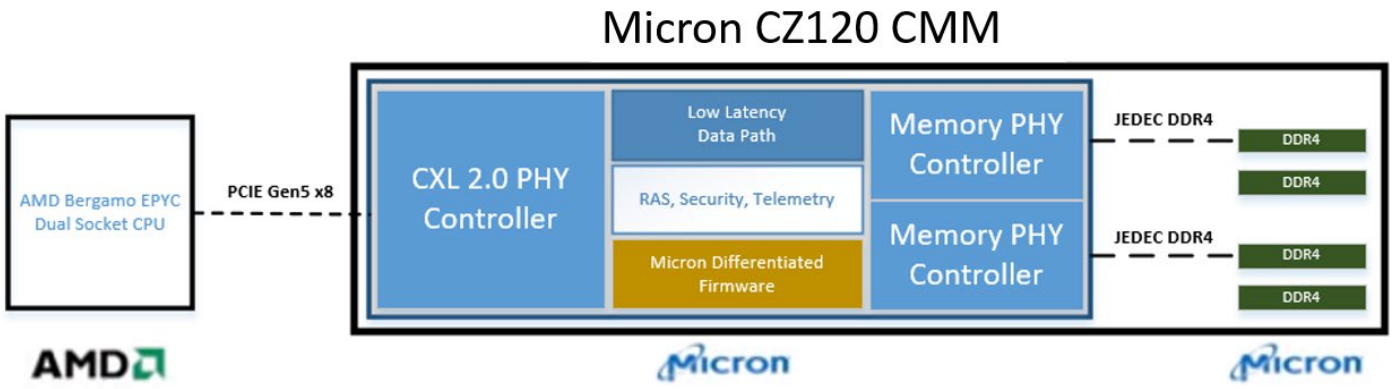


Fig. 3. AMD platform and Micron CZ120 CMM high-level product details.

Software and Hardware Support for CXL

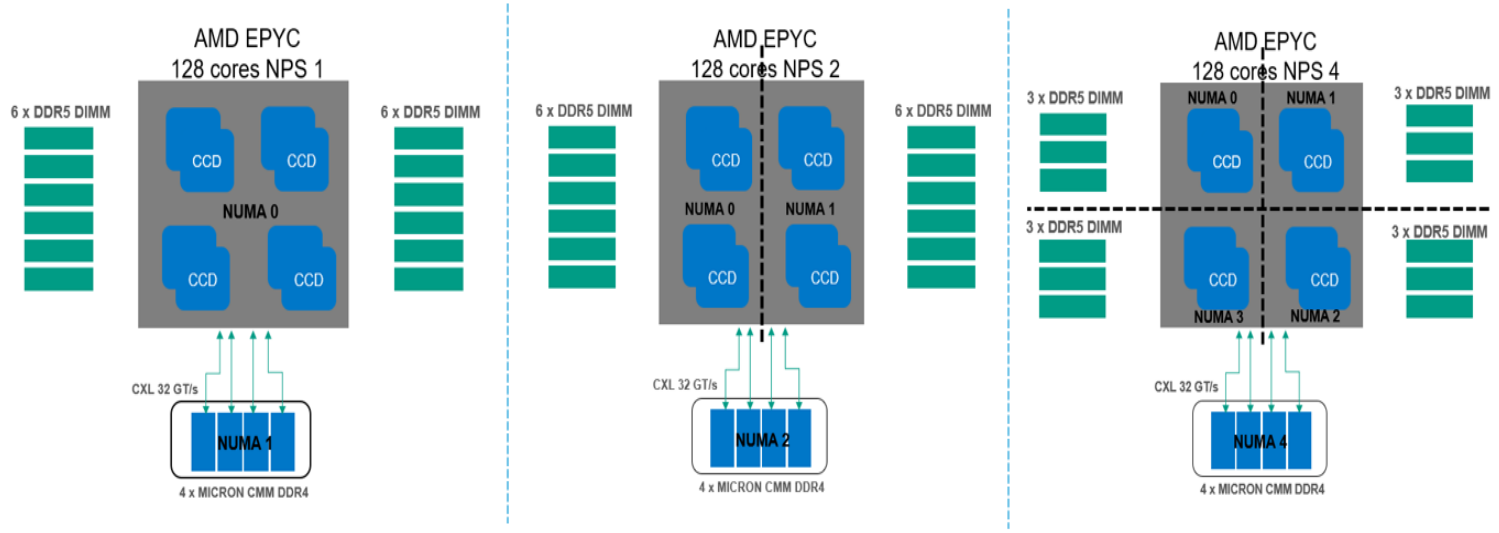
To achieve the optimal application performance, the memory hierarchy strategy focuses on storing "hot" memory regions (frequently reused data) closer to the CPU in main memory, while "cold" memory regions (infrequently reused data) are relegated to storage media, which is typically larger but slower. However, the process of moving data between main memory and storage media incurs a significant performance penalty due to the substantial latency gap between the two. This creates an opportunity for a new tier of memory (far memory) consisting of higher capacity and comparable access latency to the near memory. Because of such characteristics, CXL can be placed between main near-memory and storage media, expanding the memory hierarchy. Ideally, the goal is to store "hot" data in the near main memory, "warm" data on CXL, and "cold" data on storage to optimize application performance. The "hotness" of data is typically determined by some measure of its frequency, recency, or reuse in a particular workload.

System software developers, especially in the Linux community, have been making important advancements in improving the performance of applications running across NUMA domains and memory tiers [2]. An AMD Bergamo platform equipped with AMD EPYC 9754 processors offers wide configurability of NUMA domains. It supports the concept of NUMA node per socket (NPS) to improve the performance of different workloads. Supported NPS settings follow:

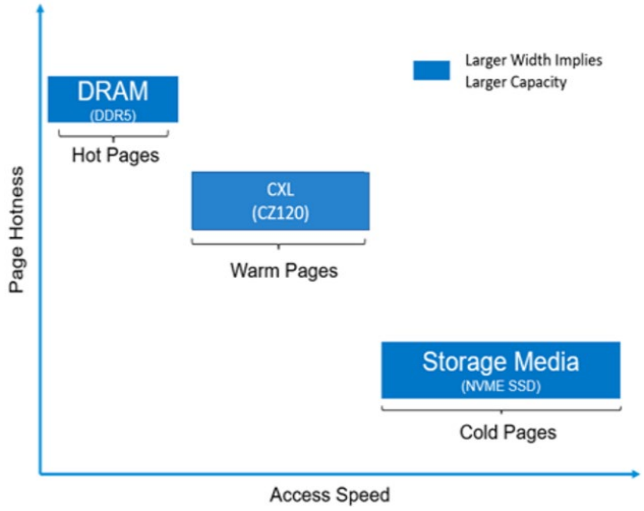
- **NPS1**—Each socket is in a single NUMA domain with all the cores in the socket and its associated memory connected to the socket in one NUMA domain and all the CMMs forming another NUMA domain.
- **NPS2**—Each socket is divided into two NUMA domains with 6 memory channels connected to each NUMA domain and all CMMs forming the third NUMA domain.
- **NPS4**—Each socket is partitioned into four NUMA domains. Each NUMA domain has three memory channels, and the memory access is interleaved across these three memory channels in each domain. All CMMs are interleaved together forming the fifth NUMA domain.

As shown in Fig. 4, CXL-based memory expanders appear in Linux as a new memory-only or CPU-less NUMA node.

In this paper, we examine various value propositions of CXL Memory Expansion within the context of the NUMA configurations mentioned above. Specifically, it analyzes the implications of different settings, namely NPS1, NPS2, and NPS4, for software-based heterogeneous interleaving. These settings enable software page-level interleaving so that pages can be allocated for a specific workload to be distributed between local DRAM and CXL NUMA nodes at ratios of 50% DRAM/50% CXL (NPS1), 66% DRAM/34% CXL (NPS2), and 80% DRAM/20% CXL (NPS4). Further details regarding the practical utilization of these settings are elaborated on in the CloverLeaf analysis section.



(a)



(b)

Fig. 4. a) AMD EPYC and Micron CMM cards divided into NUMA domains: (i) NPS1, (ii) NPS2, and (iii) NPS4. b) Memory tiering as page hotness and access speed of media.

In the context of memory capacity expansion through memory tiering, the NPS1 configuration is a logical choice. In this configuration, each CPU socket, along with its local DDR modules, constitutes the first NUMA domain, while all four CMMs together form the second NUMA domain. The NUMA domains are established based on access latency, with the local DRAMs connected to the CPU socket forming the faster “near” NUMA domain, and the CXL memory serving as the comparatively slower “farther” NUMA domain due to its higher access latency. Existing operating systems like Linux already possess the capability to perform NUMA tiering/balancing. This allows for the placement of hot pages (frequently accessed) into the closer NUMA domain (local DRAM) and warm pages (moderately accessed) and cold pages (infrequently accessed) into the farther NUMA domain (CXL), and the coldest pages into the memory with the highest latency access (typically SSDs).

In short, CXL Memory Expansion is readily supported with the existing hardware and software stack without requiring any changes to the existing setup.

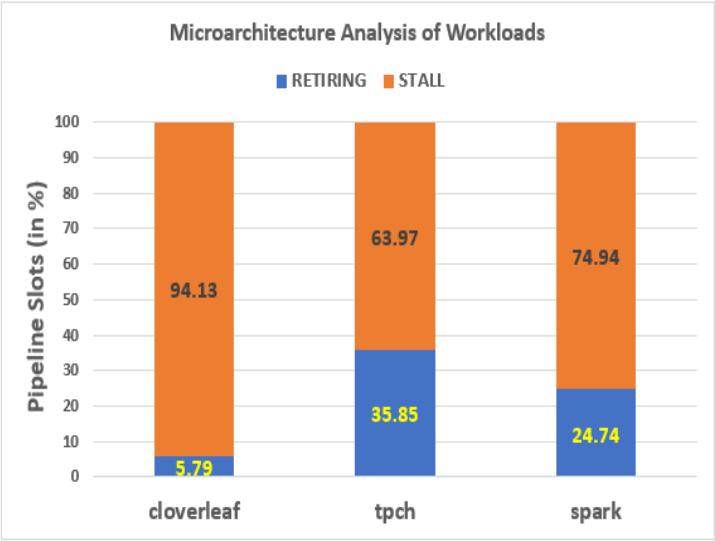
Workload Benchmark Description

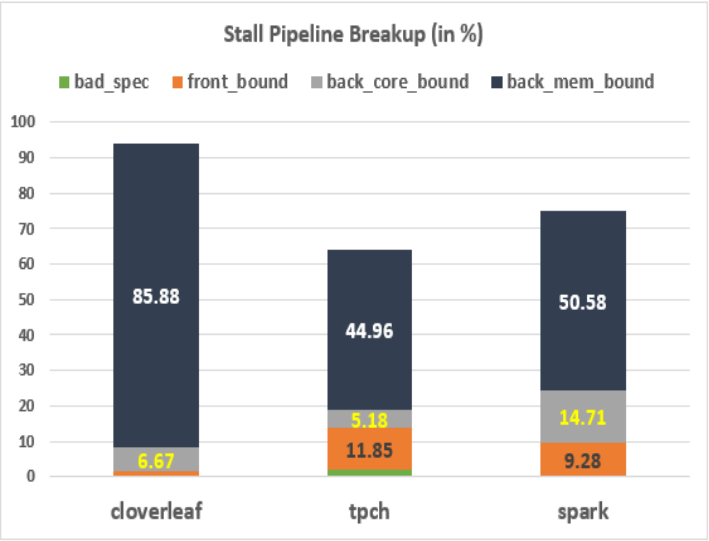
The following workloads have been chosen to assess the value proposition of CXL-attached DRAM in terms of overall system performance. These workloads are sensitive to different metrics such as capacity, latency, and bandwidth. Fig. 5 presents a microarchitecture top-down analysis of these workloads, as reported by the AMD uProf profiler toolkit [3].

- **MSSQL + TPC-H™ (OLAP)**—A database software running an online analytical processing (OLAP) benchmark suite. It responds to business-related queries and performs analysis on a large volume of data stored in data warehouse [4]. The top-down microarchitecture analysis (see Fig. 5) shows that TPC-H workload stalls 1/3 of the CPU pipeline slots and most part of the stalls are backend memory bound. Since the workload consumes only 50% of the total DRAM bandwidth available in the system, it is not bottlenecked by bandwidth and is more sensitive to memory latency than bandwidth. As this workload deals with a large amount of data and requires multiple parallel tasks for online query processing, additional memory capacity, as provided by CXL, is valuable and helps scale the application execution to high core counts while minimizing IO stalls.

- **CloverLeaf (High Performance Computing)**—A mini-app that solves the compressible Euler equations on a Cartesian grid using an explicit second-order accurate method [5]. Each cell stores three values: energy, density, and pressure, and a velocity vector is stored at each cell corner. A microarchitecture analysis of this workload in Fig. 5 highlights that almost ~95% of CPU pipeline is stalled and almost all of it is backend memory bound. Further, memory bandwidth utilization data shows that the workload consumes almost the entire DRAM bandwidth. This indicates that CloverLeaf is a highly bandwidth sensitive workload.

- **Apache Spark™ Based Machine Learning Support Vector Machine (SVM) (Big Data Workload)**— Apache Spark is an open-source data processing framework for big data and machine learning workloads. Spark primarily offers a resilient distributed data set (RDD) as its core abstraction. An RDD is a partitioned collection of elements spread across the computing nodes, enabling parallel operations on the data. A Spark machine learning (ML) SVM is used for training and regression on ML data sets [6]. The top-down analysis of the workload (Fig. 5) shows that it is a memory-bound workload and sensitive to memory latency and memory capacity when processing large data sets.





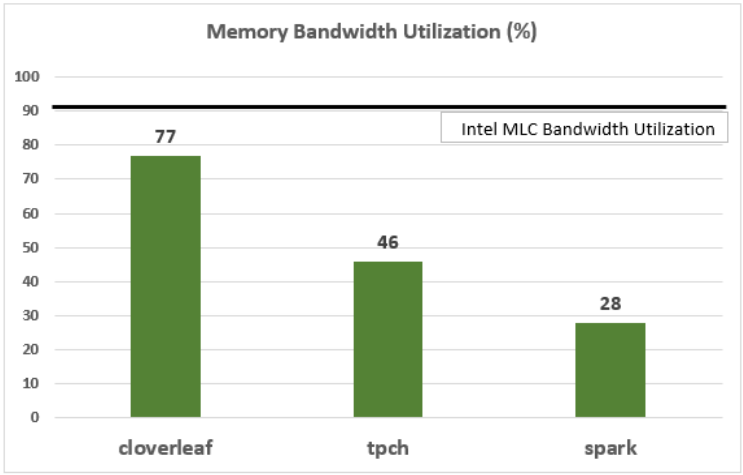


Fig. 5. Microarchitecture analysis and bandwidth utilization analysis of different workloads.

Workload Performance Analysis

The hardware configuration used for running the workloads is reported in Table I.

Table I
Hardware Configuration for Running Workloads

	Hardware Setup
Processor	AMD EPYC 9754 128-core ('Bergamo')
Memory	768GB DRAM–Near (Tier 1) memory (12 x 64GB Micron DDR5 DIMMs) or 1152GB DRAM–Near (Tier 1) memory (12 x 96GB Micron DDR5 DIMMs) 1024GB–Far (Tier 2) memory (4 x 256GB Micron CZ120 CMMs)
Storage	8x Micron 7450 NVME SSD

The performance analysis and results for the workloads are described in detail in the following subsections, including the actual memory expansion strategy used for each workload. As a general methodology, each run collects the workload performance metric for DDR5's only configuration (768GB or 1152GB) and then repeats the same experiment on DDR5 + 4 CMM devices (768GB or 1152GB contributed from local DDR and 1,024GB contributed from CMMs). Experiments were repeated at least three times for each configuration data point, and the median of those values were considered for the results.

MSSQL + TPC-H (Online Analytical Processing)

Experimental Setup

The TPC-H workload runs on an MSSQL database with a 3TB data set (3,000 scale factor) stored in column-wise format. The data set was generated by the HammerDB benchmarking software [8]. The workload was executed using the NPS1 setting, where the local memory channels were considered as one NUMA domain, and the four CMMs formed another NUMA domain. During the execution of this workload, no changes were made to the application software. The workload was executed while adhering to best practices for performance on the Linux operating system, as outlined in [9]. These best practices ensure optimal performance and efficiency during the execution of the workload on the Linux platform.

To assess the performance of the TPC-H workload, the *query throughput* metric is used. Throughput, reported as the number of queries completed within an hour, measures the system's capacity to handle a high volume of workload

requests effectively. The results for a single stream and multiple streams (total of 8 streams) are reported. Each stream is run in parallel and consists of running a set of 22 queries sequentially, following a specific query order for each stream given by the TPC-H standard specification [4].

Key Findings for Power Test Single Stream with 1 NVME Drive

Fig. 6 reports the result of a single stream, as a single power test for TPC-H running on an MSSQL server for either local DDR only or local DDR + CXL (Memory Expansion) for the configuration with 64GB memory modules. The y-axis shows the normalized query speed-up, and the x-axis represents the number of streams.

The data shows 70% speed up in single stream for DDR + CXL configuration when compared against the DDR-only configuration and 500% speed up in eight stream configurations, which seems too good to be true. After further verification, the reason was attributed to single NVME SSD bandwidth saturation in case of DDR-only system. The NVME SSD used in the system has a max sustainable bandwidth of 8GB/s. Due to which DDR performance is getting throttled, the data show that just one stream of this data set is sufficient to saturate the NVME SSD bandwidth. So, the DDR-only configuration, which has more access to NVME SSD due to demand paging is showing a much worse performance compared to a local DDR + CXL configuration. The single NVME SSD cannot satisfy the bandwidth requirements of demand paging from DDR memory.

To remove the bottleneck of the NVME SSD bandwidth from the equation, the system configuration has been updated with 8 NVME SSDs, and subsequent results have been generated.

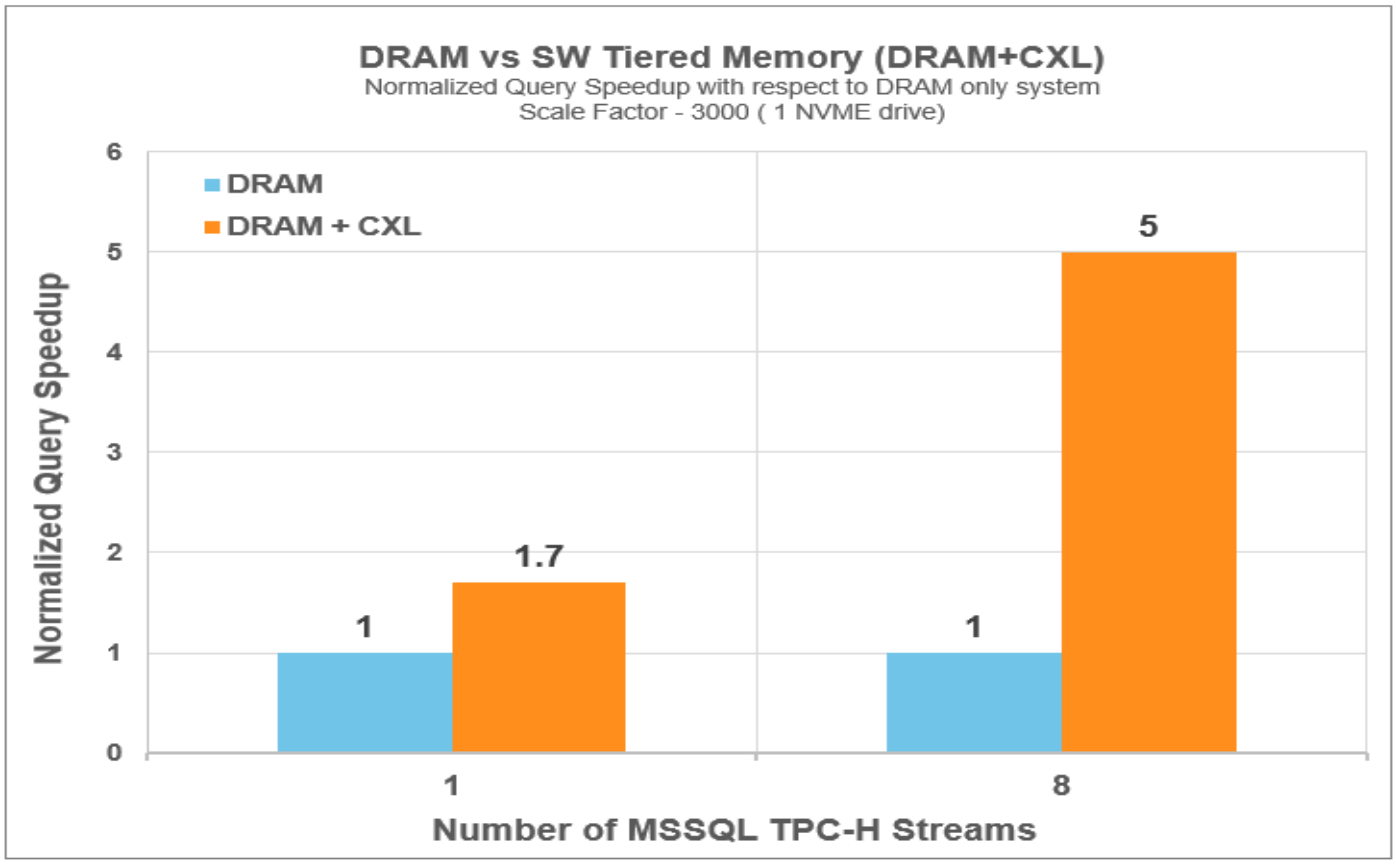


*Fig. 6. MSSQL + TPC-H power test data with 12 * 64GB DRAM and 1x4 NVME SSD (~60MB/s per core).*

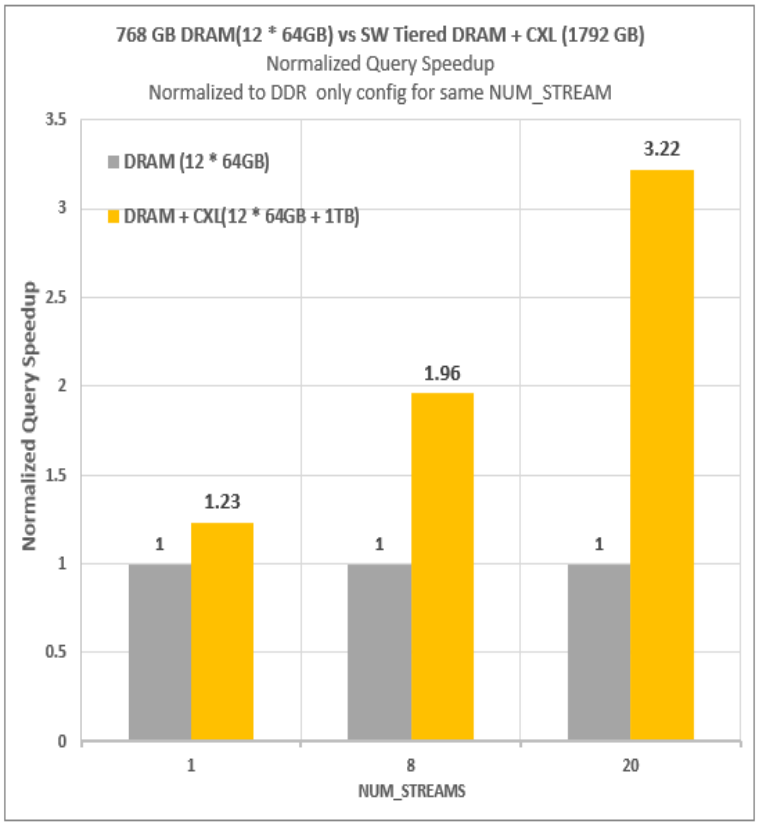
Key Findings for Multi-Stream with Multiple NVME Drives with 64GB Local DDR Modules

Fig. 7a highlights the query processing speed-up factor relative to same number of streams with 8 NVME SSD's and 64GB Local DDR modules. The speed-up factor and amount of IO transaction reduction for DDR-only setup (768GB) for different number of parallel workload streams is compared against the same number of streams run with DDR + CXL setup (1,792GB). In both setups, the workload size is greater than the system memory size. In Fig. 7a, as the number of streams

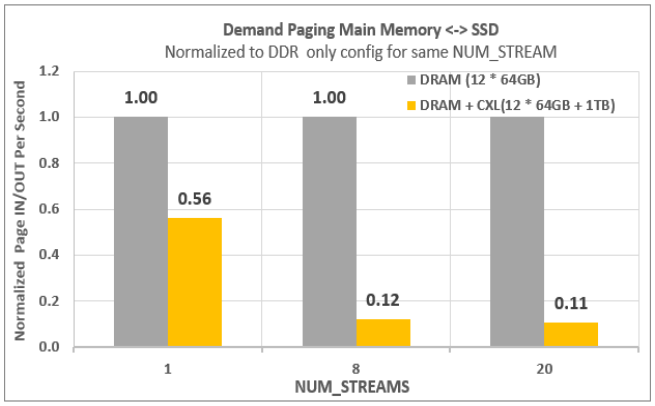
increases from 1 to 20, the DDR + CXL setup query execution speed shows an increase of ~3.2x relative to the DDR-only setup for the same number of workload streams. For a single stream test, the improvement is lower at ~1.23x. Fig 7b shows that with the additional memory in the form of CXL, the number of IO transactions has dropped by 44% for a single-stream test and 88% for an 8-stream test, relative to local DDR-only system configuration. This also highlights the key reason for the 1.96x improvement in query speed-up for the 8-stream workload.

Fig. 7c shows the amount of memory used in different setups. The normalized memory footprint ratio measures the ratio between the amount of memory used by the workload for the DDR + CXL setup and DDR-only setup. When considering a single stream, a modest increase of 42% in the memory footprint brings about a notable reduction of 44% in IO transactions. This reduction ultimately leads to a 23% improvement in query speed-up. In the case of 8 and 20 parallel workload streams, an increase of 128% and 141% in the memory footprint brings about a reduction of 88% and 89% in IO transactions respectively. This reduction leads to a 96% and 222% improvement in query speed-up for 8 and 20 streams of TPC-H workload respectively.

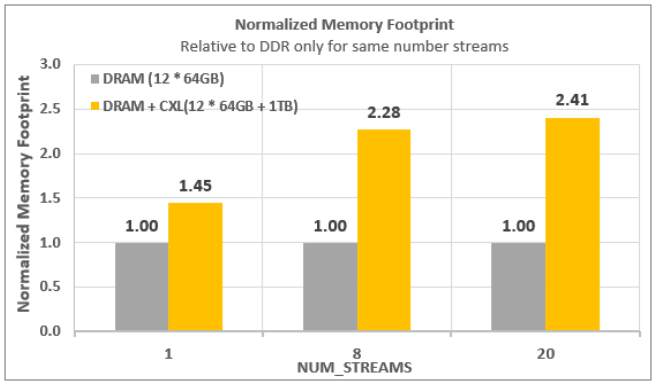
Fig. 8 highlights the same datapoint as Fig.7a with just one modification i.e. the execution speedup for different parallel stream runs are normalized by the results obtained from a single stream run on local DRAM (different normalization reference point). Representing the data in this form allows us to understand the saturation point while scaling up the workload. As we can see from Fig.8, DDR only config peaks in performance at 8 parallel streams of workload. Any subsequent increase in number of streams results in degraded performance due to memory contention. Software tiered memory setup comprising of DRAM and CXL allows us to scale up to 16 parallel streams of TPC-H workload after which the performance saturates and no performance gain is seen hereafter with increased workload streams. Memory contention effect happens after 24 streams. Fig.8 also establishes that the key reason for a huge 3.22x jump in performance relative to DRAM setup for 20 streams as seen in Fig 7a is mainly due to the drop in performance for 20 streams in DRAM only setup. DRAM only setup peaks in performance at 8 parallel streams. The peak to peak peformance delta between DRAM only(8 streams) and DRAM + CXL(16 streams) setup is seen to be at 2.3x.



(a)



(b)



(c)

*Fig. 7. Multi-stream TPC-H test run on MSSQL server with 12 * 64GB DRAM modules and multi-SSD (>300MB/s per core): (a) normalized query speed-up, (b) normalized IO transactions and (c) normalized memory footprint.*

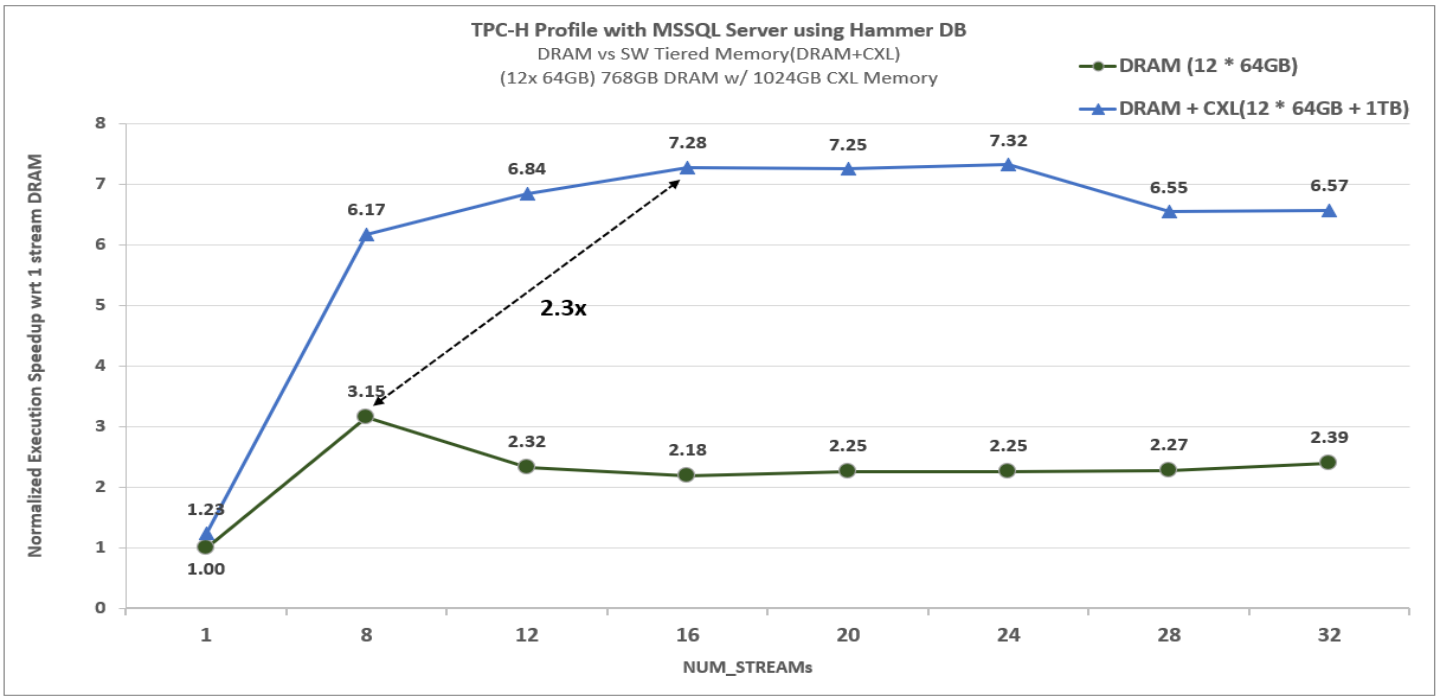


Fig. 8. Multi-stream TPC-H Query Execution Speedup relative to single stream run on DRAM only setup (12 * 64GB) and multi-SSD (>300MB/s *per* core)

Key Findings for Multi-Stream with Multiple NVME Drives with 96GB Local DDR Modules

Fig. 9 highlights the query processing speed-up factor versus the number of streams with 8 NVME SSD bandwidth and 96GB Local DDR modules. The speed-up factor and amount of IO transaction reduction for DDR-only setup (1152GB) is compared against the DDR + CXL setup (2,176 GB). In both setups, parallel multiple streams of the workload are run to ensure that the aggregate dataset size is greater than the system memory size. In Fig. 9, as the number of streams increases from 1 to 20, the DDR + CXL setup query execution speed shows the best relative execution speedup at 20 streams with 69% improvement, relative to the DDR-only setup for the same number of workload streams. For a single stream test, the improvement is lower at ~1.19x. Fig 9b shows that with the additional memory in the form of CXL, the number of IO transactions has dropped by 70% for an 8-stream test and 83% for 20 stream test, relative to local DDR-only system configuration. This also highlights the key reason for the 23% and 69% improvement in query speed-up for 8-stream and 20-stream workload.

Fig. 9c shows the amount of memory used for different setups. The normalized memory footprint ratio measures the ratio between the amount of memory used by the workload for the DDR + CXL setup and DDR-only setup. When considering 8 parallels streams, an increase of 71% in the memory footprint brings about a notable reduction of 70% in IO transactions. This reduction ultimately leads to a 23% improvement in query speed-up. In the case of 20 parallel streams, an increase of 89% in the memory footprint brings about a reduction of 83% in IO transactions. This reduction leads to a 69% improvement in query speed-up for the TPC-H workload relative to performance seen by running the same number of streams on DDR only setup.

We would like to draw your attention to single stream data showing 19% query speedup with only 11% IO reduction vs 8 streams of workload showing only 23% query speedup with a 70% reduction in IO operations, relative to native DDR configuration for same number of workload streams. While the numbers are collected and averaged over multiple runs ensuring the authenticity of the data, it is hard to establish exact reasoning in some of these corner cases since the performance of the workload varies significantly with the memory capacity breakup between native DDR and CXL, workload footprint and number of parallel streams.

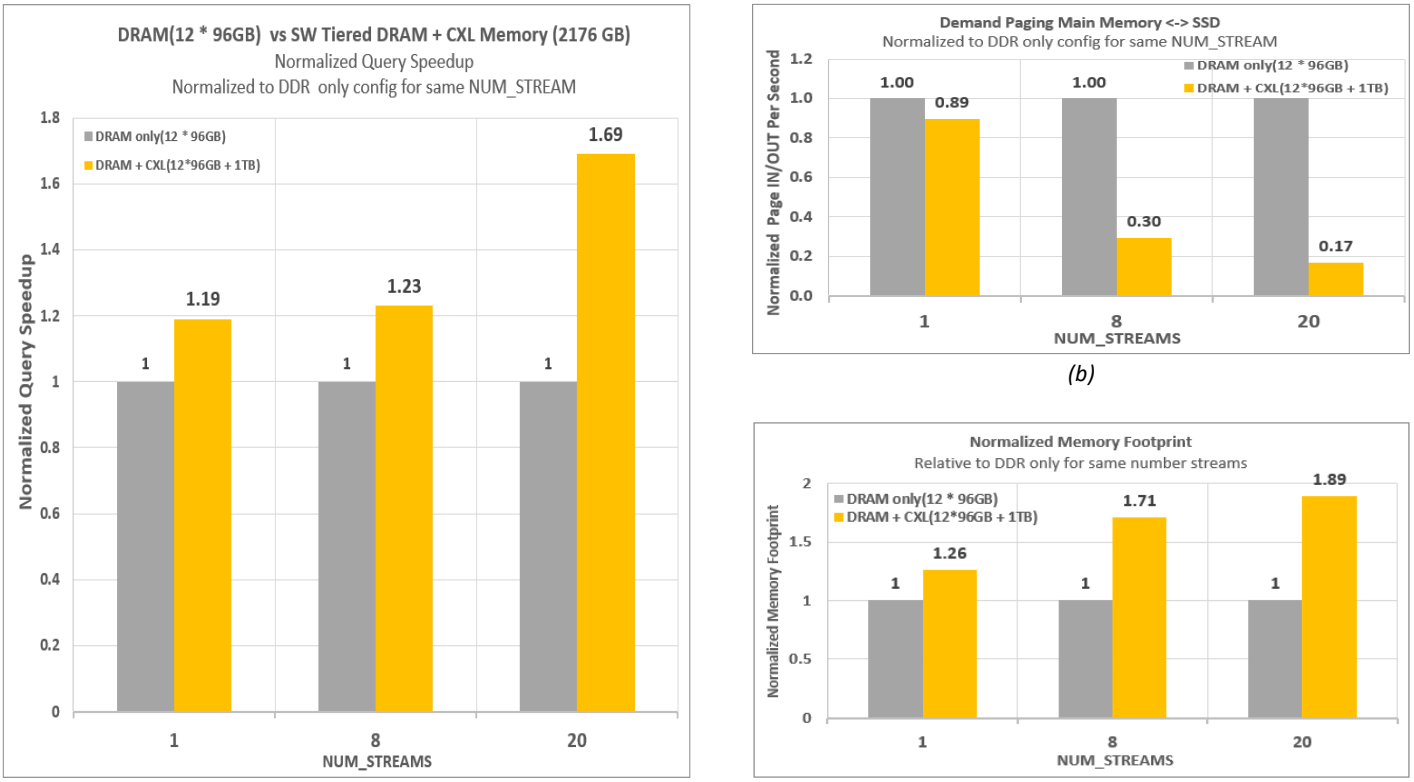


*Fig. 9. Multi-stream TPC-H test run on MSSQL server with 12 * 96GB DRAM modules and multi-SSD (>300MB/s per core): (a) normalized query speed-up, (b) normalized IO transactions and (c) normalized memory footprint.*

Fig. 10 highlights the same datapoint as Fig.9a with one modification i.e. the execution speedup for different parallel stream runs are normalized to speedup value obtained from single stream run on local DRAM setup (different normalization reference point). Representing the data in this form helps us to evaluate the scale-up value proposition when we introduce CXL memory. As seen in Fig.10, DDR only config peaks in performance at 8 parallel streams of TPC-H workload. The execution speedup relative to single-stream run on local DDR stays constant after peaking at 8 parallel streams. Software tiered memory setup comprising of DRAM and CXL allows us to scale up to 28 parallel streams of TPC-H workload after which the performance saturates.

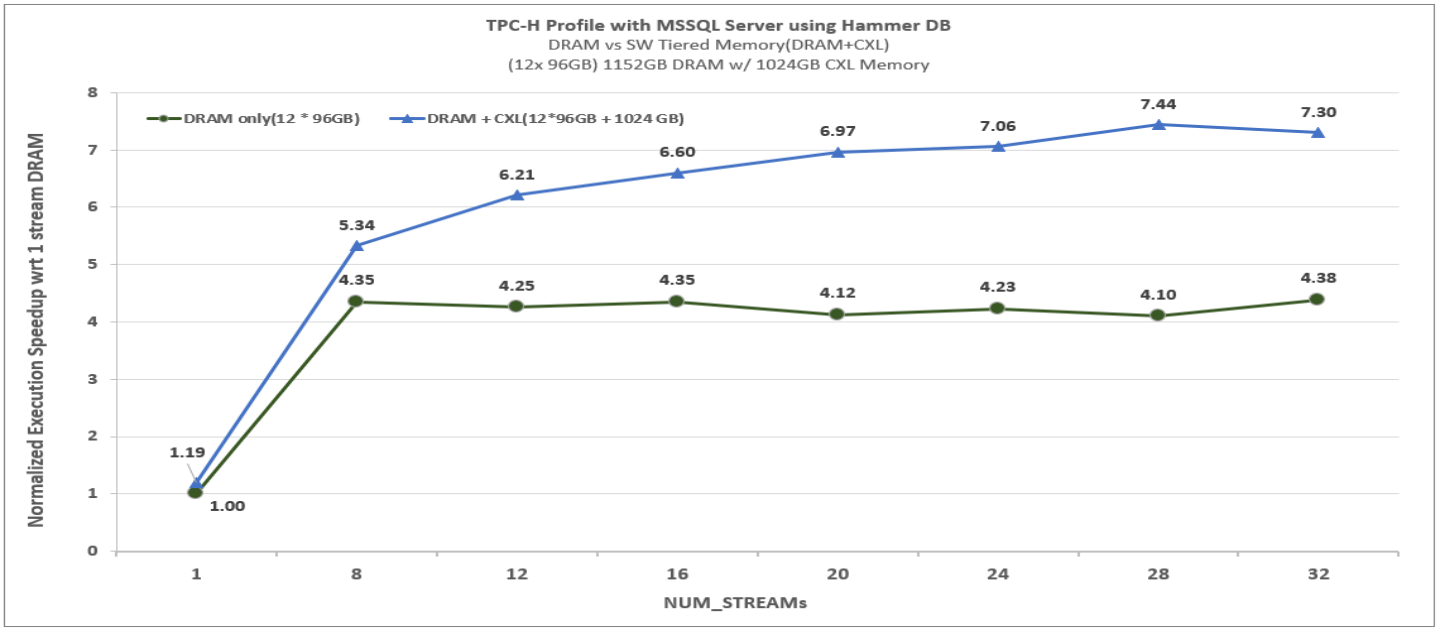


*Fig. 10. Multi-stream TPC-H Query Execution Speedup wrt to single stream run on DRAM only setup (12 * 96GB) and multi-SSD (>300MB/s per core)*

Final Summary

Fig. 11 compares the execution speedup for multi-stream with 64GB DDR modules, 96GB DDR modules, 64GB DDR modules with 1 TB of CXL memory and 96GB DDR modules with 1 TB of CXL memory, relative to the query speedup observed with single stream workload run on 64GB DDR modules only setup. The data helps us in establishing the fact that instead of upgrading from 64GB to 96GB local memory DDR modules, CXL can help in capacity expansion and lower the total cost of ownership. Furthermore, the addition of CXL cards allows us to scale up more effectively compared to DDR only configs of 96GB modules. This can be verified by the fact that DRAM + CXL using 64GB DDR modules for its local DRAM configuration has 1.6 times higher performance than DRAM only configuration using 96GB DDR modules and saturates in query speedup performance at 16 or 20 parallel workload streams as against 8 streams of TPC-H workload running in parallel.

As seen from Fig.11, both DDR only configs peak in query speedup at 8 parallel MSSQL TPC-H streams whereas 64 GB DDR modules with CXL (total capacity = 1796 GB) peaks in query speedup at 16 streams and 96 GB DDR modules with CXL (total capacity = 2192 GB) peaks in query speedup at 28 parallel TPC-H streams.

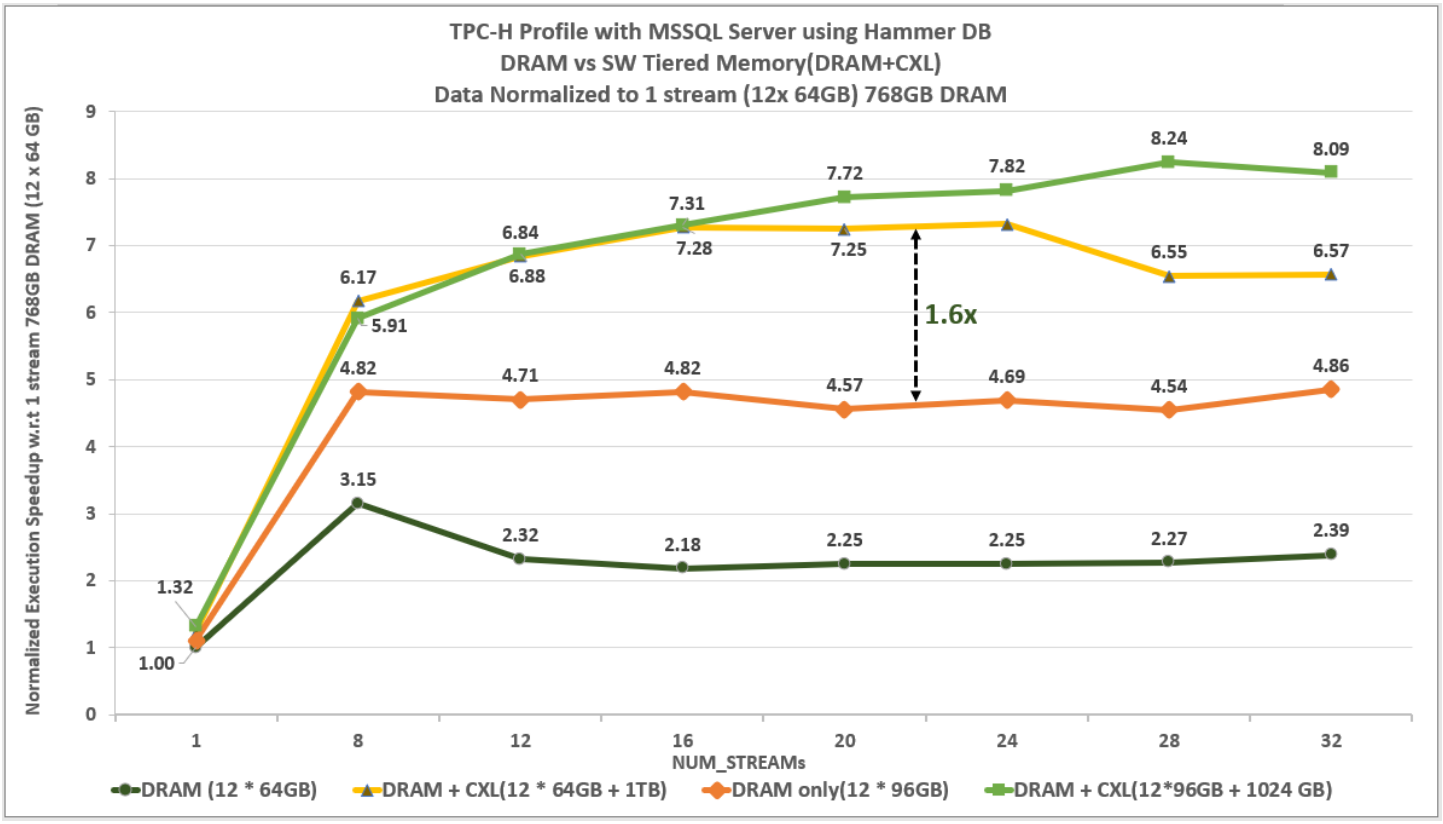


*Fig. 11. Multi-stream TPC-H test run on MSSQL server - query speed-up for different system memory configs relative to single stream DRAM only config (12 * 64GB)*

CLOVERLEAF (High-Performance Computing)

Experimental Setup

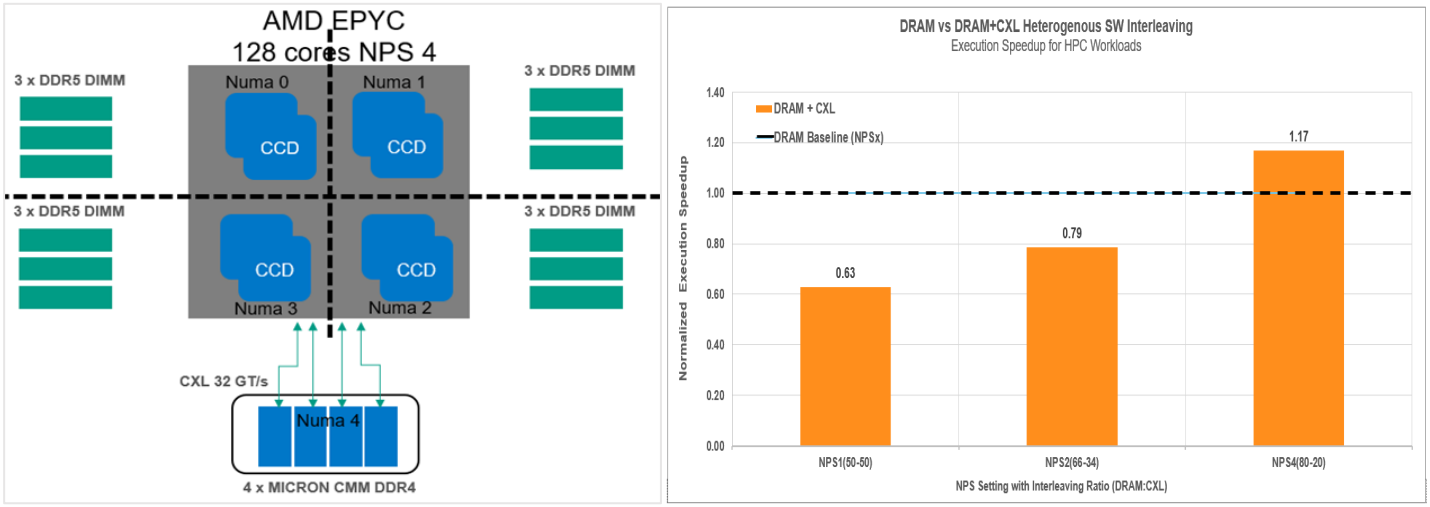
This workload was run with NPS1, NPS2, and NPS4 settings and with all CMMs forming a single NUMA domain in each of these cases. The input used was “clover_bm1024_short.in” from the official distribution deck. The workload is run via “numactl” Linux command performing a round-robin interleaving policy. Consequently, this allows for an interleaving ratio of 50%-50% (NPS1), 66%-34% (NPS2) and 80%-20% (NPS4) where the first number is the amount of physical memory pages allocated to local DDR and the second is the number of physical memory pages allocated to CXL memory.

Key Findings

For CloverLeaf, the CMMs were configured to provide bandwidth expansion. For such workloads, the best performance is observed when an AMD EPYC 9754 based platform having 12 memory channels of 64GB memory module clocked at 4800 MT/s is configured to a NPS4 configuration through BIOS. Using NPS4, the local CPU socket is divided into four NUMA domains with each NUMA domain having three local DDR channels interleaved together. The four CXL channels form the fifth NUMA domain. This ensures 80% of memory allocations are on the local DDR channels and 20% on the CXL channels. When only local DDR channels are used (without CXL), 100% of the memory allocations are performed on the local DDR channels with NPS4 configuration. This is a form of software interleaving using heterogenous memories.

Results in Fig. 12b show how software heterogenous memory interleaving helps in improving performance for HPC workloads. The data shows 17% speed-up for 12 channels of 64GB DDR+ 4 CMMs when compared to 12 channels of 64GB DDR only when interleaving is in the ratio of 80:20 where 80 corresponds to the percentage of physical pages mapped to local DDR setup and remaining 20% mapped to the CMMs. As established in microarchitecture top-down analysis of workloads in Fig. 4, CloverLeaf is a bandwidth intensive workload. With the addition of 4 CMM, there is a bandwidth expansion of 33% relative to 12 DDR channels at 4800 MT/s. The additional idle latency penalty of CXL compared to native DRAM is compensated for by the bandwidth expansion provided by multiple CMMs. The improvement in speed-up of CloverLeaf by 17% clearly demonstrates the value proposition of bandwidth expansion of CXL using software heterogenous interleaving.

The results will be the same even if the local memory module is increased to 96GB from 64GB since HPC workloads such as Cloverleaf are not sensitive to memory capacity.



(a) *(b)*

Fig. 12. a) NPS4 architecture for DDR + CXL setup. b) CloverLeaf execution speed-up with SW defined page allocation ratio.

SPARK ML SVM (Big Data Analytics)

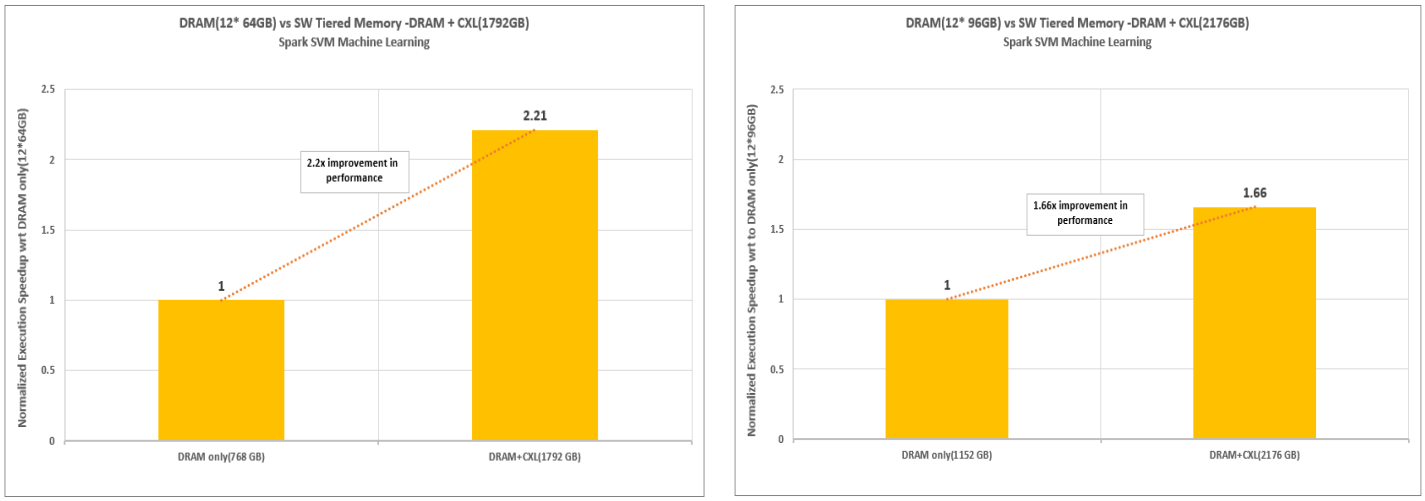
Experimental Setup

This workload was run with an NPS1 configuration targeting capacity expansion via memory tiering. The raw data set is 360GB, but Spark's memory footprint grows during data processing and needs memory for intermediate processing steps, taking advantage of memory sizes larger than that of DDR-only setup.

Key Findings with Multiple NVME Drives and 64GB/96GB Local DDR Module

According to Fig. 13a, the incorporation of 1 TB CXL as a new memory pool managed through memory tiering leads to a significant 2.21x enhancement in execution speed-up compared to a configuration solely reliant on 768 GB local DRAM. When the same workload was run on 1152 GB local DRAM with 1 TB CXL memory, the execution speedup is seen to be

1.66x relative to 1152 GB of local DRAM. With the ability to persist crucial data structures such as RDDs in memory, Spark leverages the expanded memory facilitated by CXL to enable efficient reuse of data during parallel operations. This enhancement ultimately enhances overall data processing performance. Caching an RDD is a mechanism to accelerate applications that repeatedly access the same RDD. Without caching or checkpointing, an RDD is re-evaluated every time an action is triggered on that RDD.



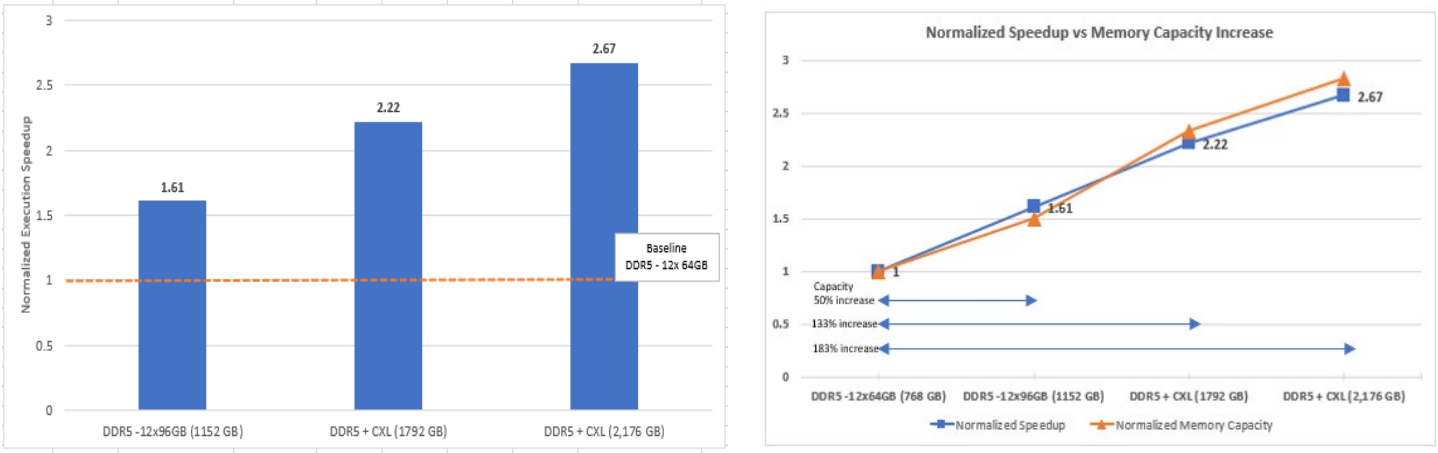
(a) *(b)*

Fig. 13. Spark ML Execution Speed-up for DDR + CXL relative to DDR only configuration a) 786 GB DDR & 1TB CXL b) 1152 GB DDR & 1TB CXL

Final Summary

Spark ML workload is highly sensitive to incremental memory capacity. As seen in Fig 14a, different memory capacity i.e., 1152GB (12 * 96GB Local DRAM), 1756GB (12 * 64GB Local DRAM + CXL), 2176 GB (12 * 96GB Local DRAM + CXL), shows a 1.61x, 2.22x and 2.67x improvement in execution speedup relative to 768 GB Local DRAM as baseline.

To understand the workload's sensitivity to incremental memory capacity contributed through either CXL cards or larger density local DRAM (96GB), the normalized execution speedup and memory capacity increase relative to 768 GB local DRAM as shown in Fig 14b. The data shows that both the execution speedup and memory capacity increase overlap each other leading to the conclusion that Spark SVM is less sensitive to increased access latency of CXL memory and workload performance improves linearly with the increase in capacity. This trend is not observed on MSSQL TPC-H.



(a) *(b)*

Fig. 14. (a) Spark ML Execution Speed-up (b) Normalized Speedup vs Increase in Memory Capacity, relative to 768 GB DDR only configuration.

Conclusion

The performance study of different workloads for two value propositions of CXL–Memory Bandwidth Expansion and CXL–Memory Capacity Expansion leads us to the following conclusions.

CXL–Memory Bandwidth Expansion

Bandwidth sensitive workloads, such as CloverLeaf, spend most of the CPU pipeline slots in stalls, and more than >75% these stalls are backend memory bound. CXL memory can help in bandwidth expansion using software-based interleaving between DDR and CXL memory. Results on CloverLeaf show that when systems are tuned appropriately, the workload can benefit up to 17% when CXL expands the bandwidth by 33%.

While the results show performance speed-up of 17% relative to local DDR-only configuration, this improvement is seen only with NPS4. NPS1 and NPS2 on the other hand show performance degradation relative to only a local DDR config. The bandwidth ratio between local DRAM operating at 4800 MT/s and 4 CMMs is ~2.8x. If these bandwidth ratios change due to an increased speed of local DDR, a different number of CXL versus local DDR channels, the interleaving ratio also needs to change.

CXL–Memory Capacity Expansion

Workloads such as Spark ML SVM and TPC-H running on MSSQL generate a lot of IO transactions due to demand paging from DDR. CXL when introduced as a tiered memory to local DDR helps in significantly reducing the number of IO transactions. Warm data is placed in CMM, closer to local DDR with latency much less than accessing IO media.

TPC-H single stream analysis shows that with a mere 42% memory expansion relative to local DDR-only config comprising of 64GB modules, IO transactions are reduced by 44% resulting in a performance speed-up of 23%. TPC-H 8-stream analysis shows that with 128% memory footprint expansion, IO transactions come down by 88%, resulting in a performance speed-up of 96%. DDR only configs saturate in throughput at 8 parallel multi-streams of TPC-H workloads. A similar trend is seen when 96GB DDR only modules are used for the experiment. However, adding CXL memory improves the performance relative to either of the DDR only configs and helps in scaling up number of parallel streams of workload. The increase in performance is not linear to the increase in memory capacity. This clearly demonstrates that as we reduce our transactions to storage media, capacity- or latency-sensitive workloads can get a huge boost in performance. Similar trends were seen for Spark SVM workload as well with one distinction. Spark SVM execution speedup was found linear to increase in memory capacity.

It is important to understand that different workloads have different characteristics and sensitivity to metrics such as latency, bandwidth, and capacity. The system configuration such as memory tiering or memory interleaving serves to extract the best value proposition for the different workloads.

References

[1] Pond: CXL-Based Memory Pooling Systems for Cloud Platforms (https://arxiv.org/abs/2203.00241)
[2] TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory (https://arxiv.org/pdf/2206.02878.pdf)
[3] AMD uProf: https://www.amd.com/en/developer/uprof.html
[4] TPC-H Homepage: https://www.tpc.org/tpch/
[5] CloverLeaf Benchmark: https://www.amd.com/en/developer/zen-software-studio/applications/spack/cloverleaf.html
[6] Apache Spark SVM: https://spark.apache.org/docs/latest/ml-classification-regression.html#linear-support-vector-machine
[7] Intel MLC Documentation : https://www.intel.com/content/www/us/en/developer/articles/tool/intelr-memory-latency-checker.html
[8] HammerDB benchmarking tool: https://www.hammerdb.com/
[9] SQL Server Linux Performance Best Practices: https://learn.microsoft.com/en-us/sql/linux/sql-server-linux-performance-best-practices?view=sql-server-ver16