

The Advantages of Managing the CXL Memory Tier in Hardware

Intel® Flat Memory Mode on the Latest Generation of Intel® Xeon® 6 Processors to Support CXL-Based Memory Tiering

Intel Authors

Ivan Cuevas

Escareno

Systems Performance
Engineer

Swadesh

Choudhary

Principal Engineer

Thomas Willhalm

Senior Principal
Engineer

Anil Godbole

Senior Product
Planning & Marketing
Manager

Mark Adams

Strategic
Communications Lead

Abstract

In-memory databases dominate today's data center workloads. The large memory footprints of these databases necessitate bigger server memory capacities. However, memory is the highest cost item in a data center server bill of materials, leading to higher server costs.

New generations of multi-core processors are available to support the computing power needed. Besides the native DRAM memory channels, these processors support CXL interfaces to augment the memory capacity of the server using CXL Type-3 devices¹. CXL-based memory tiering can be an effective strategy for enhancing the memory-to-core ratio in the server. O/S or software-based tiering techniques have been introduced in recent years to support CXL-based memory tiering. The Intel® Flat Memory Mode (FMM) feature on the latest generation Intel® Xeon® 6 processors offers an innovative hardware-based technique to do the same.

This paper focuses on Intel Flat Memory Mode as a hardware-based memory tiering technique and explores its performance for in-memory database management compared to O/S-based software techniques.

Table of Contents

I: Introduction	1
II: CXL Memory Tiering.....	2
III: Intel Flat Memory Mode (Hardware-controlled CXL Memory Tiering)	3
IV: In-memory Database Workload Measurements and Comparison	4
V: Conclusion.....	6
Authors	6
References	7

I: Introduction

In-memory databases are central to many of today's data center workloads. They can be used for caching data to speed up e-commerce servers or in stand-alone database applications like those required for running Enterprise Resource Planning (ERP) applications.

As the scope of web-based applications continues to grow, so do the memory footprints of the underlying in-memory databases. However, memory is the highest cost item in a data center server bill of materials, leading to higher server costs.

New generations of multi-core CPUs are available to support the computing power needed. Besides the native DRAM memory channels, these CPUs support CXL interfaces to augment the memory capacity of the server using CXL Type-3 devices. CXL-based memory tiering can be an effective strategy for enhancing the memory-to-core ratio in the server. O/S or software-based tiering techniques have been introduced in recent years to support CXL-based memory tiering. The Intel Flat Memory Mode feature on the latest generation of Intel® Xeon® 6 processors offers an innovative hardware-based technique to do the same.²

CXL-based memory can reduce the cost of memory on a server. It allows for the use of lower cost memory DIMMs on both native DRAM and CXL channels to realize the required memory capacity. This is less expensive than using higher priced/higher capacity DIMMs on the native DRAM channels alone to meet the same capacity.³

TCO savings aspects of CXL memory, however, is not the focus of this paper. We will focus on Intel Flat Memory Mode as a hardware-based memory tiering technique and explore its performance for In-memory database management compared to O/S-based software techniques. Performance is evaluated using the open-source database MongoDB, in contrast to previous work that relied on closed-source database.⁴

Section II describes the essential elements of CXL memory tiering used in O/S (or software) based techniques like AutoNUMA,⁵ or TPP (Transparent Page Placement).⁶ Special frameworks like MEMTIS,⁷ SMT,⁸ HMSDK,⁹ HybridTier,¹⁰ or NeoMem¹¹ are beyond the scope of this paper, however.

Section III describes the Intel Flat Memory Mode feature available on the latest-generation (Generation 6) Intel Xeon processors for doing hardware-controlled CXL memory tiering.

Section IV describes the experimental setup used to perform database testing and compares the results for both software-based techniques (AutoNUMA, TPP) and hardware-based Intel Flat Memory Mode. We chose the MongoDB database with YCSB (Yahoo cloud-based benchmarking tool) available in the public domain for our measurements.

Section V includes the Conclusion section of our findings.

II: CXL Memory Tiering

In this section, we will describe the basic software-based techniques used for performing CXL Memory Tiering. The two most common utilities available for doing this in Linux kernels (since v5.18) are AutoNUMA and Transparent Page Placement (TPP). A server fitted with CXL memory boots up with total available memory in two separate NUMA node tiers. The native DRAM memory (Near' Memory) node and the CXL-attached memory (Far' memory) node. When a workload gets assigned to any memory from the Far memory tier it will experience higher latency when accessing data from that tier leading to degradation in performance. The goal of memory tiering is to hide this additional latency.

The Linux operating system implements this by migrating page contents from Far (CXL) memory to the Near Memory (native DRAM) if it determines that the page in Far Memory has become 'hot' - meaning a predetermined threshold of frequency of accesses to this page has been crossed. The workload is typically stalled when doing the migration. However afterwards, any new accesses by the workload to this memory page are from the Near memory tier thereby avoiding the additional latency of CXL.

This assumes that there is sufficient space in Near Memory to do the migration. If not, the O/S must find some 'cold' pages - pages whose frequency of accesses has fallen below a certain threshold - in the Near memory and migrate them to Far (CXL) memory to make room. The graphic in Figure 1 shows this process.

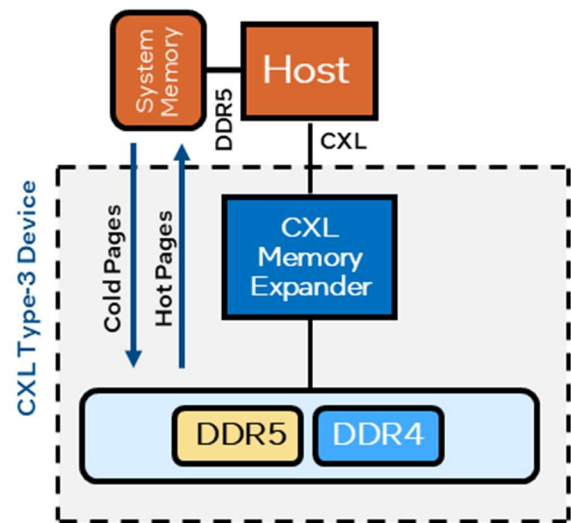


Figure 1: Expanding System Memory with CXL Memory Devices

Inherent to the memory tiering techniques are the requirements on the O/S for:

- 1) Performing Hot/Cold page telemetry on the total server memory space.
- 2) Performing page migrations, temporarily stalling affected workloads.

Modern data center servers typically support memory capacity of hundreds of gigabytes to a few terabytes. Hence, the associated page tables (assuming a 4K byte page size) are quite large. They are implemented as multilayer structures to reduce the overhead on a server's memory capacity. For doing the Hot/Cold page telemetry, the O/S has to dedicate some CPU H/w resources - typically one core from a multi-core processor. The telemetry involves continually scanning the multi-layer page table to create the needed statistics to establish which entries (pages) are hot and which are cold.

For doing page migrations, the O/S will typically deploy a data-movement accelerator, e.g. the Intel® Data Streaming Accelerator (Intel® DSA) on the Intel Xeon processor family.¹² If such offload engines are not in use, the data movement is implemented using Programmed I/O, which results in longer workload stalls.

As explained above, in software-based memory tiering methods, the O/S requires using certain hardware resources on the CPU which otherwise would be available to execute client workloads.

III: Intel Flat Memory Mode (Hardware-controlled CXL Memory Tiering)

While there have been multiple variants of hardware memory tiering, Intel Flat Memory Mode is the industry's first cache-line granular, hardware managed memory tiering solution for CXL.¹³ Intel Flat Memory Mode offers a two-tier solution with a 1:1 ratio between the two tiers (Figure 2). For this section, locally attached DRAM is referred to as "Near Memory" or "NM" and the CXL attached memory is referred to as the "Far Memory" or "FM." In this mode, hardware manages data placement between the two tiers without any intervention from software. The aggregate capacity of locally attached DRAM and CXL attached memory is exposed to software as a unified memory range and data is placed exclusively in one

of the tiers only. Note that this is different from previous hardware managed memory tiering solution for Intel® Optane™ persistent memory¹⁴ which was known as Intel Memory Mode – where DRAM served as an inclusive cache for non-volatile memory.

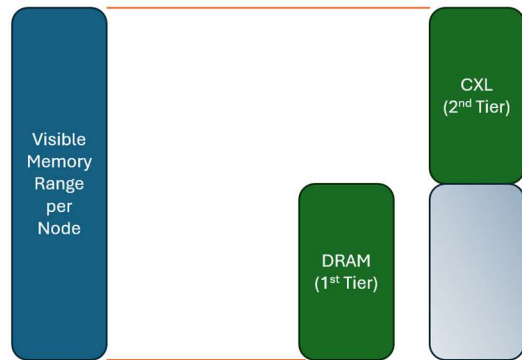


Figure 2: Intel Flat Memory Mode Concept

Intel Flat Memory Mode's 1:1 ratio means that the DRAM and CXL memory capacity used for tiering are equal, i.e. if Near memory's total capacity is 1TB, the Far memory capacity is also 1TB. Thus, the configuration in this example has a total memory capacity of 2TB.

Hardware uses one bit of metadata to manage the location of data between the two tiers (thus the metadata overhead for this feature is very low as well).

The 1:1 ratio only requires 1b of metadata, often referred to as the "tag," representing two states:

- 1'b0 - Data is at home
- 1'b1 - Data is not at home

The "Home" is an important concept in this mode. The NM component of memory is in the system address space, denoted as address A, and its corresponding FM location as address B (address B = address A + NM capacity). The home for address A is NM and the home for address B is FM. For example, let's assume DRAM NM capacity is 24GB and hence CXL FM is also 24GB for the 1:1 ratio. The total address space exposed to the O/S-software is from 0 - 48GB. If address A = 4GB, its corresponding address B = 28GB. After initialization, the system starts with zeroed NM and FM and the tag bit in NM is 0, meaning dataA is at home (in NM). A read request to address 28GB is sent to NM. The memory controller reads the NM location and decides it is a miss. The memory controller then sends a fill-read request to the FM (system address B, 28GB) location to get dataB requested by the read. DataB is filled into DDR NM and data A now is evicted to the FM location. The tag in NM is updated to 1'b1, indicating dataA is no longer at home. Figure 3 illustrates this swap, for brevity, the metadata value updates are not shown. To reduce local memory misses and improve the performance of workloads with cache-unfriendly memory access patterns, Intel Flat Memory Mode supports adding dedicated local DRAM that is not hardware-tiered as a separate range in the physical memory address space. This dedicated local memory is exposed as a second NUMA node alongside the first NUMA node which contains the hardware tiered memory.

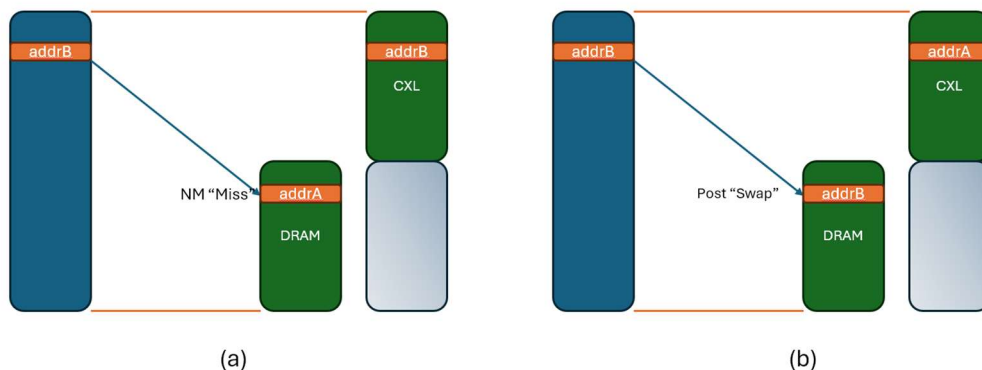


Figure 3: Illustration of Data Being Swapped from Far Memory (CXL) to Near Memory (Native DRAM)

IV: In-memory Database Workload Measurements and Comparison

We evaluated Intel Flat Memory Mode on current Intel Xeon 6 processors with P-cores. The workload under evaluation is MongoDB, benchmarked using the Yahoo! Cloud Serving Benchmark (YCSB).¹⁵ The study characterizes the performance of three memory tiering strategies—Intel Flat Memory Mode (FMM) with Intel’s CXL-based hardware tiering, Linux AutoNUMA, and Transparent Page Placement (TPP) – and compares them against a baseline DRAM-only, non-tiered configuration. By analyzing these approaches side by side, we provide insight into how hardware-assisted tiering stacks up against open-source, software-driven alternatives in practical database scenarios.

The system under test was a 2-socket Intel® Xeon® 6767P processor (formerly codenamed Granite Rapids), each socket providing 64 cores with Intel® Hyper-Threading Technology enabled, for a total of 128 vCPUs per socket. The system was running Ubuntu 24.04 LTS with 6.12.0 kernel. For our experiments, two different memory configurations were evaluated:

Baseline (DRAM-only) Configuration: Each socket was equipped with 8* 128 GB DDR5-6400 DIMMs in a 1 DIMM-per-channel (1DPC) layout, yielding 1 TB of DRAM per socket as depicted in Figure 4.

Baseline:

Socket 0	1024GB native DRAM
Socket 1	1024GB native DRAM

Figure 4: Memory Setup for Baseline Configuration

Tiered Configurations (FMM, AutoNUMA, TPP): Each socket used 8* 64GB DDR5-6400 DIMMs in 1DPC, providing 512GB of DRAM. This was combined with two x16 CXL DDR5 devices, each populated with four 64GB DDR5-4800 DIMMs,

for an additional 512GB of CXL-attached memory per socket. In total, this configuration also provided 1TB of memory per socket, split evenly between DRAM and CXL as shown in Figure 5.

Tiered Memory and Intel Flat Memory Mode:

Socket 0	512GB native DRAM	512GB CXL
Socket 1	512GB native DRAM	512GB CXL

Figure 5: Memory Setup with Tiered Memory

For simplicity, the MongoDB server was pinned to Socket 0, while the YCSB client ran on Socket 1. This setup ensured that all communication flowed over the UPI links, avoiding the complexity of a separate network configuration. Socket 0 hosted 8 instances of MongoDB Enterprise v7.0.17, each pinned to 16 vCPUs. Socket 1 ran 8 YCSB client instances, each configured with 25 million operations of Workload 9010 (90% reads, 10% updates) and using a different number of query threads per client.

To evaluate different levels of data tiering, we configured the workload across four record count profiles, each designed to represent a specific data footprint relative to system memory:

Size	Memory footprint	# of records	Memory utilization
S	500GB	40M	~50%
M	700GB	60M	~75%
L	860GB	80M	~90%
XL	900GB	96M	~95%

Our YCSB benchmark evaluation demonstrates that Intel Flat Memory Mode achieves performance parity with traditional DRAM-based configuration. See Figure 6. Testing across multiple MongoDB footprint sizes reveals that Intel Flat Memory Mode delivers throughput performance within 1-3% of pure DRAM implementations, while maintaining miss rates up to 11% in the largest DB size (~900GB).

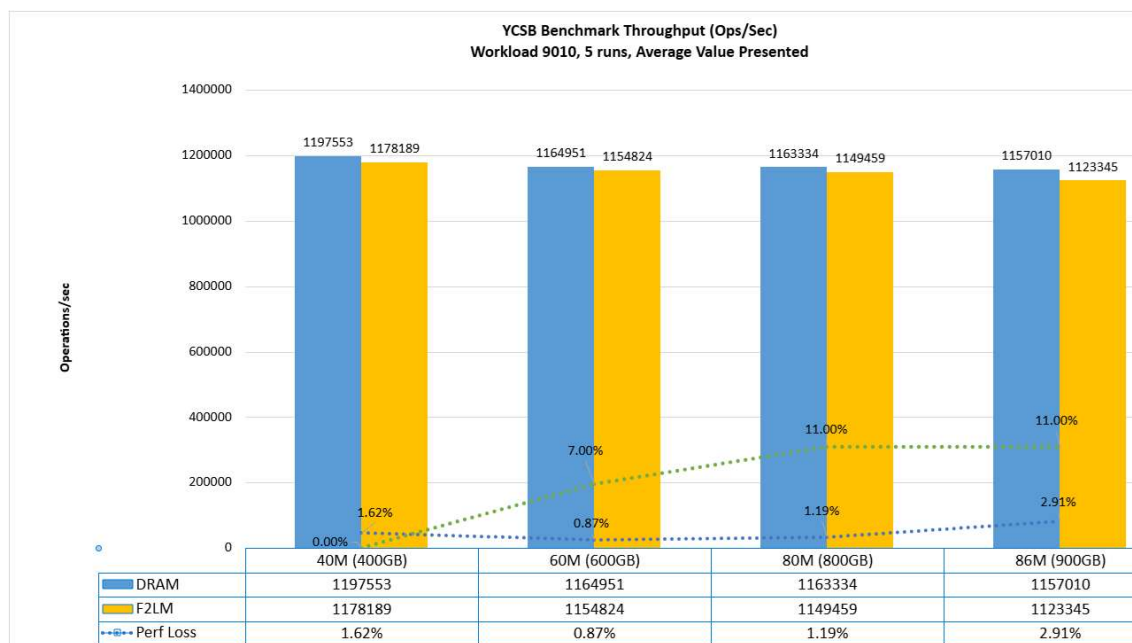


Figure 6: TYCSB Benchmark Achieves Almost the Same Performance with Intel Flat Memory Mode

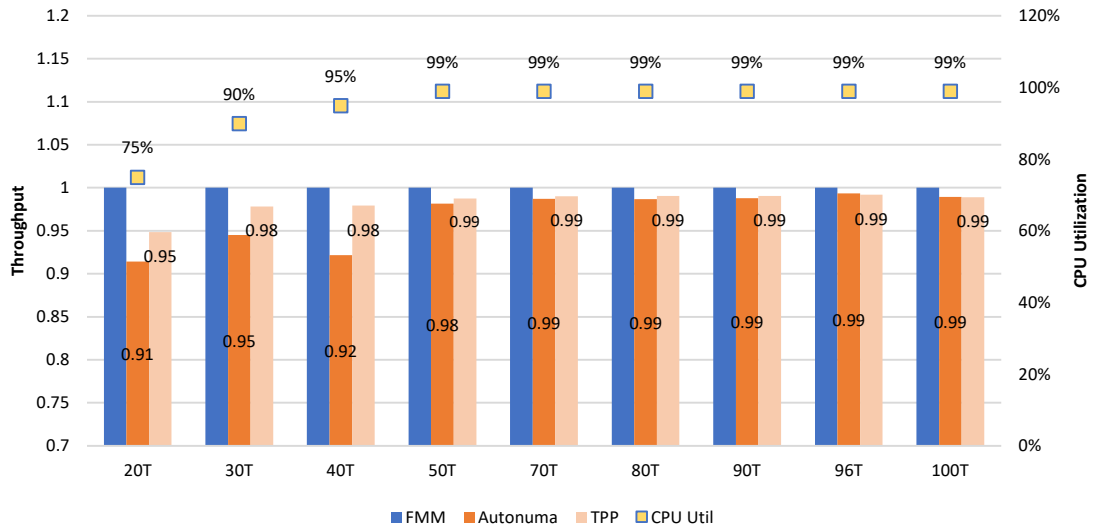


Figure 7: Intel Flat Memory Mode Brings Up to 9% Better Performance than Software Tiering Under Full Memory Utilization

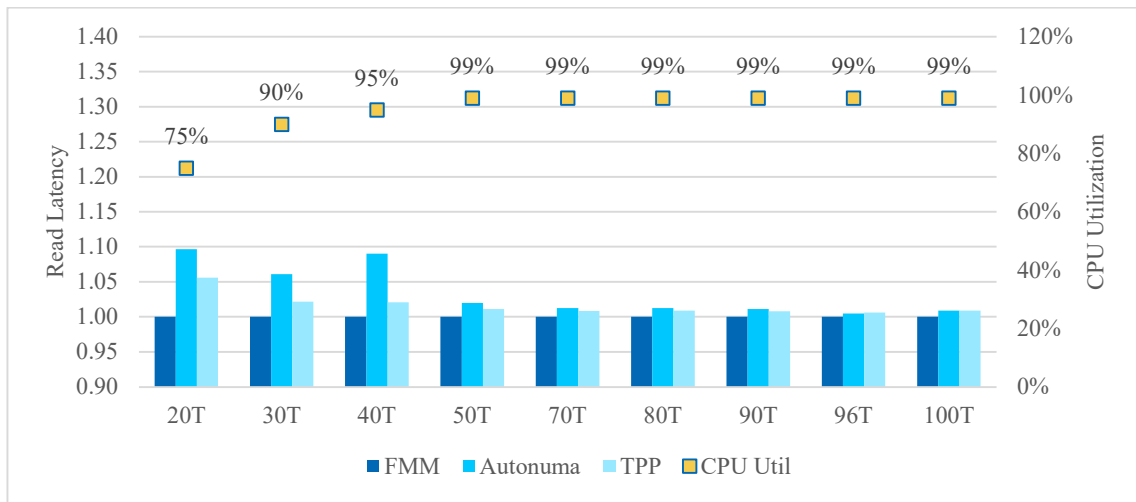


Figure 8: YCSB Read Latency and CPU Utilization for Different Load Levels

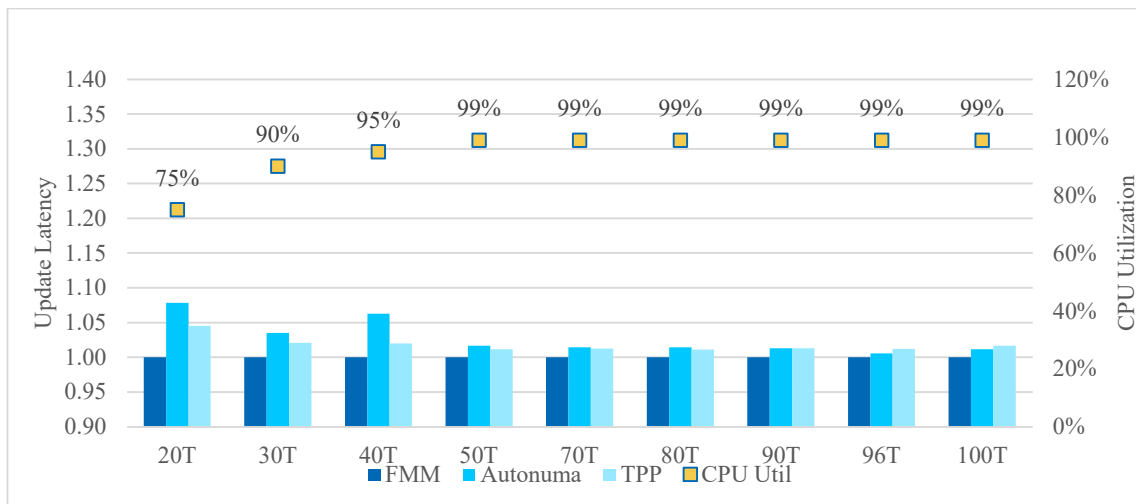


Figure 9: YCSB Update Latency and CPU Utilization for Different Load Levels

Our YCSB threading scalability evaluation demonstrates that Intel Flat Memory Mode delivers higher throughput compared to software-based memory tiering approaches. We tested YCSB query threads ranging from 20 to 100 across 8 MongoDB instances, each processing 86 million records. The results highlight the benefits of hardware-based tiering compared to software-based tiering, particularly under moderately high CPU utilization (75%), where it achieved 9% and 5% higher throughput compared to AutoNUMA and TPP, respectively. Refer to [Figure 7](#).

Our analysis also demonstrates that Intel Flat Memory Mode delivers superior low-latency performance compared to software-based memory tiering approaches across different CPU load levels under different database transactions, as can be observed in read requests in [Figure 8](#) and update requests in [Figure 9](#). This demonstrates that the latency advantages of Intel Flat Memory Mode do not come at the cost of computational efficiency, providing both performance and resource optimization benefits.

V: Conclusion

This technical paper presents a comprehensive evaluation of Intel Flat Memory Mode as a hardware-based memory tiering solution for CXL type 3 enabled systems, specifically examining its performance characteristics for in-memory database workloads compared to DRAM-only and software-driven tiering solutions.

Our results show that Intel Flat Memory Mode delivers performance parity with a DRAM-only configuration, with throughput just 1–3% lower across various database footprint sizes, this while maintaining a Near Memory miss rate below 11% even in the largest database scenarios. Intel Flat Memory Mode demonstrates superior scalability by outperforming software-driven tiering solutions, delivering 9% higher throughput compared to AutoNUMA and 5% higher throughput compared to TPP under moderate to high CPU utilization scenarios. The hardware-managed approach provides superior latency characteristics across different load levels without compromising computational efficiency, making it particularly well-suited for latency-sensitive database applications. This performance advantage stems from Intel Flat Memory Mode's transparent cache-line granular migrations with minimal metadata overhead, whereas software-driven tiering solutions cause workload stalls during page migrations.

Authors



Mark Adams is a Strategic Communications Lead in the Data Center Group at Intel Corporation. He focuses on customer communications related to quality and reliability.



Ivan Cuevas Escareno is a Systems Performance Engineer at Intel Corporation, focusing on performance optimization across large-scale platforms. His work involves evaluating next-generation memory and interconnect technologies to improve efficiency and scalability. He received his B.S. degree in Computer Science from the University of Guadalajara, Mexico. His research interests include emerging memory technologies such as Compute Express Link (CXL) and MRDIMMs, and their use in tiered memory solutions.



Swadesh Choudhary is a Principal Engineer at Intel Corporation, Santa Clara, CA, 95052, USA. His research interests include PCI-Express, CXL, Universal Chiplet Interconnect Express (UCIe), and their standardization and implementation. Choudhary received his M.S. degree in electrical and electronics engineering from Stanford University. He is a Senior Member of IEEE.



Anil Godbole is a Sr. Product Planning & Marketing Manager at Intel. His current focus is on Intel's Data Center CPU CXL strategy. He is also the Marketing Chair of the Compute Express Link consortium. He has 25+ years of technology industry experience.



Thomas Willhalm is a Senior Principal Engineer working at Intel since 2005, specializing in enterprise software and databases, computer architecture and memory systems. Thomas has a diploma in mathematics from the University of Constance and a Ph.D. in computer science from the TU Karlsruhe (KIT). His experience includes the use of SIMD instructions and persistent memory in databases. Among other areas, his current work focuses on in-memory databases and advanced memory technologies including CXL.

References

- ¹ CXL 3.2 Specification. (2024) Compute Express Link Consortium. <https://computeexpresslink.org/cxl-specification/>
- ² Gianos, Chris. "Architecting for flexibility and value with next gen Intel® Xeon® processors." 2023 IEEE Hot Chips 35 Symposium (HCS). IEEE Computer Society, 2023. J. Baik, J. Kim, C. H. Park and J. Ahn, "Accelerating Page Migrations in Operating Systems With Intel DSA," in IEEE Computer Architecture Letters, vol. 24, no. 1, pp. 37-40, Jan.-June 2025, doi: 10.1109/LCA.2025.3530093.
- ³ R. Hayden and P. Schell, "Opportunities and challenges for compute express link (CXL)," ABI research, Tech. Rep., 2024.
- ⁴ M. Ahn, T. Willhalm, D. Lee, N. May, J. Kim, D. Ritter, and O. Rebholz, "Exploiting locality in flat memory with cxl for in-memory database management systems," in Proceedings of the 21st International Workshop on Data Management on New Hardware, ser. DaMoN '25. New York, NY, USA: Association for Computing Machinery, 2025. <https://doi.org/10.1145/3736227.3736230>
- ⁵ D. Moura, D. Mossé and V. Petrucci, "Performance Characterization of AutoNUMA Memory Tiering on Graph Analytics," 2022 IEEE International Symposium on Workload Characterization (IISWC), Austin, TX, USA, 2022, pp. 171-184, doi: 10.1109/IISWC55918.2022.00024.
- ⁶ Hasan Al Maruf, Hao Wang, Abhishek Dhanotia, Johannes Weiner, Niket Agarwal, Pallab Bhattacharya, Chris Petersen, Mosharaf Chowdhury, Shobhit Kanaujia, and Prakash Chauhan. 2023. TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory. In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS 2023). Association for Computing Machinery, New York, NY, USA, 742–755. <https://doi.org/10.1145/3582016.3582063>
- ⁷ Z. Zhou *et al.*, "NeoMem: Hardware/Software Co-Design for CXL-Native Memory Tiering," 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO), Austin, TX, USA, 2024, pp. 1518-1531, doi: 10.1109/MICRO61859.2024.00111.
- ⁸ K. Kim *et al.*, "SMT: Software-Defined Memory Tiering for Heterogeneous Computing Systems with CXL Memory Expander," in IEEE Micro, vol. 43, no. 2, pp. 20-29, 1 March-April 2023, doi: 10.1109/MM.2023.3240774
- ⁹ K. Lee *et al.*, "Improving Key-Value Cache Performance with Heterogeneous Memory Tiering: A Case Study of Compute-Express-Link-Based Memory Expansion," in IEEE Micro, vol. 45, no. 2, pp. 102-113, March-April 2025, doi: 10.1109/MM.2024.3358861
- ¹⁰ Kevin Song, Jiacheng Yang, Zixuan Wang, Jishen Zhao, Sihang Liu, and Gennady Pekhimenko. 2025. HybridTier: an Adaptive and Lightweight CXL-Memory Tiering System. In Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '25). Association for Computing Machinery, New York, NY, USA, 112–128. <https://doi.org/10.1145/3676642.3736119>
- ¹¹ Z. Zhou *et al.*, "NeoMem: Hardware/Software Co-Design for CXL-Native Memory Tiering," 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO), Austin, TX, USA, 2024, pp. 1518-1531, doi: 10.1109/MICRO61859.2024.00111.
- ¹² "Intel® Data Streaming Accelerator User Guide" <https://www.intel.com/content/www/us/en/content-details/759709>; and "Intel® Data Streaming Accelerator Architecture Specification: <https://www.intel.com/content/www/us/en/content-details/671116>
- ¹³ Yuhong Zhong, Daniel S. Berger, Carl Waldspurger, Ryan Wee, Ishwar Agarwal, Rajat Agarwal, Frank Hady, Karthik Kumar, Mark D. Hill, Mosharaf Chowdhury, and Asaf Cidon. 2024. Managing memory tiers with CXL in virtualized environments. In Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation (OSDI'24). USENIX Association, USA, Article 3, 37–56.
- ¹⁴ "Intel Optane Data Center Persistent Memory," 2019 IEEE Hot Chips 31 Symposium (HCS), Cupertino, CA, USA, 2019, pp. i-xxv, doi: 10.1109/HOTCHIPS.2019.8875668.
- ¹⁵ Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10). Association for Computing Machinery, New York, NY, USA, 143–154. <https://doi.org/10.1145/1807128.1807152>



Performance varies by use, configuration and other factors. Learn more on the Performance Index site.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

© 2026 Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.