# CoaXiaL: A CXL-Centric Memory System for Scalable Servers

Albert Cho† Anish Saxena† Moinuddin Qureshi Alexandros Daglis
Georgia Institute of Technology

***Abstract*—The memory system is a major performance determinant for server processors. Ever-growing core counts and datasets demand higher memory bandwidth and capacity. DDR—the dominant processor interface to memory—requires a large number of on-chip pins, which is a scarce resource, thus limiting the processor's memory bandwidth. With limited bandwidth, multiple concurrent memory requests experience significant queuing delays that often overshadow DRAM's service time and degrade performance. We present CoaXiaL, a memory system design for throughput-oriented manycore servers that replaces *all* of the processor's DDR interfaces with the pin-efficient CXL interface, which offers 4× higher bandwidth per pin. While such replacement incurs a considerable latency overhead, we demonstrate that, for many workloads, and with careful integration, CXL's higher bandwidth more than offsets its latency premium. Our evaluation shows that CoaXiaL improves the performance of manycore throughput-oriented servers by 1.39× on average and by up to 3×.**

## I. Introduction

Manycore processor architectures have been delivering performance gains despite the end of Dennard scaling and Moore's law slowdown in the past two decades. To accommodate exponential data growth, technological breakthroughs have enabled higher-capacity memory with new media like non-volatile RAM (NVRAM) or via remote memory access over fast networks (e.g., RDMA), yet at significantly inferior memory access latency and bandwidth compared to the DDR-based main memory. As a result, servers continue to predominantly rely on DDR-attached memory for performance while optionally retaining a slower memory tier like NVRAM or remote DRAM for capacity expansion.

The Compute Express Link (CXL) standard aims to bridge the performance gap between low-bandwidth, high-capacity memory and DDR-based main memory. CXL enables vast expansion of a processor's memory capacity and bandwidth availability, while retaining DDR-like characteristics at a modest latency premium. Consequently, there is much interest in architecting CXL-based memory systems that leverage memory pooling and capacity expansion capabilities [3], [21], [30], [38], [57]. This paper studies how CXL, a technology based on an open standard with widespread industry adoption, can be leveraged to reshape the memory system of future scalable (i.e., manycore) server processors.

CXL's high bandwidth stems from its underlying PCI Express (PCIe) serial interface, which currently delivers about 4× higher bandwidth per processor pin compared to the parallel DDR interface, with technological roadmaps projecting this bandwidth gap to grow further. Hence, by repurposing the processor's DDR-allocated pins to CXL, it is possible to *quadruple* the available memory bandwidth, a welcome boost for throughput-oriented, bandwidth-constrained *manycore* servers. Consider modern server processors that feature 4–12 cores per memory channel. The higher-end ratio represents recent and upcoming throughput-optimized server processors with 128+ cores, such as AMD's EPYC Bergamo [42] and Intel's Granite Rapids and Sierra Forest [43]. Given rigid pin constraints, shifting from conventional DDR-centric to CXL-centric memory systems can overcome the bandwidth wall for such servers, by leveraging CXL's bandwidth-per-pin advantage.

The key drawback of bandwidth-rich CXL-centric memory systems is their associated memory access latency overhead, currently ∼50ns for directly connected (non-multiplexed) CXL memory devices [51]. Such high latency overhead, which seemingly doubles DRAM access latency of ∼50ns, has relegated CXL-related research so far to treat the technology exclusively as a memory *expansion* technique rather than a potential *replacement* of local DDR-attached memory. However, focusing on the increase in *unloaded* DRAM access latency alone paints an incomplete picture that does not accurately represent a system's *effective* memory access latency. First, in a loaded memory system, queuing delays dominate the raw memory access time. Second, on large CPU chips, on-chip time constitutes a considerable fraction of end-to-end memory access latency. We posit that ample bandwidth availability can be leveraged to offset CXL's latency premium in two ways.

First, the effective memory access latency of heavily loaded memory systems is dominated by queuing delays at the DDR memory controllers. Mitigating these queuing delays by provisioning more memory channels would require more processor pins, which are a scarce resource. Instead, the pin-efficient CXL interface can be used to *indirectly* attach more channels to the processor, thus reducing memory access latency by mitigating queuing. Second, increased bandwidth use—afforded by higher bandwidth availability—can be traded off for latency reduction. We propose mechanisms for selective "Concurrent Access of Last-Level Cache (LLC) and Memory" (CALM) to remove the LLC's latency from the critical path to memory, thus offsetting the CXL latency overhead by as much as 40%. While CALM is not fundamentally tied to CXL, it is more effective in CXL-enabled bandwidth-rich memory systems.

Given the bandwidth superiority of CXL-based memory

† Equal contribution.

SC24, November 17-22, 2024, Atlanta, Georgia, USA
979-8-3503-5291-7/24/$31.00 ©2024 IEEE

systems and the opportunities of leveraging that strength to mitigate latency concerns, we argue that *a memory system attached to the processor entirely over CXL* is a key enabler for scalable manycore server processors that deploy memory-intensive workloads. Our design, CoaXiaL, replaces all of the processor's direct DDR interfaces with CXL for a major memory bandwidth boost, which additionally affords the CALM mechanism to further reduce memory access latency.

In summary, we make the following contributions:

- We make the radical proposal of using high-bandwidth CXL as a *complete replacement* of pin-inefficient DDR interfaces to scale the memory bandwidth wall faced by manycore server processors, a shift that disrupts decades-long DDR-centric memory system design practices. By directly addressing the memory bandwidth wall, we mitigate queuing delay, consequently reducing the effective memory access latency.
- We propose selective CALM mechanisms that leverage memory bandwidth abundance to further offset CXL's latency overhead, in addition to the memory access latency gains derived from queuing delay reduction.
- Across a wide range of workloads, we show that despite its high *unloaded* memory access latency, CoaXiaL drastically reduces the *effective* memory access time in typical scenarios with a loaded memory system, yielding a 1.39× speedup on average, and up to 3×.
- We identify limitations imposed on CXL by the current PCIe standard and highlight opportunities that a revised standard could leverage, to achieve a higher average speedup of 1.52×.

*Paper outline:* §II motivates the replacement of DDR with CXL in server processors and §III highlights the opportunities of leveraging bandwidth abundance to address latency concerns. §IV introduces the design of CoaXiaL, a new CXL-based memory system for scalable servers. We outline our methodology in §V and analyze performance results in §VI. Finally, §VII covers related work and §VIII concludes.

## II. Background

### A. Low-latency DDR-based Memory

Servers predominantly access DRAM over the Double Data Rate (DDR) parallel interface. The interface's processor pin requirement is determined by the width of the data bus, command/address bus, and configuration pins. A DDR4 and DDR5 [26] interface is 288 pins wide. While several of those pins are terminated at the motherboard, most of them (160+ for an ECC-enabled DDR4 channel [25], and likely more for DDR5 [49]) are driven to the processor chip.

The DDR interface's 64 data bits directly connect to the processor and are bit-wise synchronous with the memory controller's clock, enabling a worst-case unloaded access latency of ∼50ns. Scaling a DDR-based memory system's bandwidth requires either clocking the channels at a higher rate, or attaching more channels to the processors. The former entails signal integrity challenges [44] and a reduction in supported ranks per channel, limiting rank-level parallelism and memory capacity. The latter requires more on-chip pins, which cost significant area and power, and complicate placement, routing, and packaging [67]. Therefore, the pin-count on processor packages has only been doubling about every six years [55].

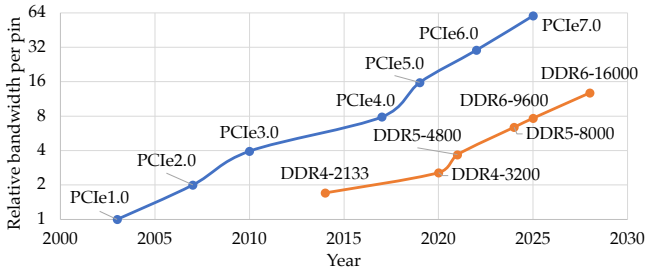



Fig. 1: Bandwidth per processor pin for DDR and CXL (PCIe) interface, norm. to PCIe-1.0. Note the y-axis log scale.

Given these pin limitations and the steady growth of core counts on high-end server processors, reducing the number of cores that contend over a memory channel is difficult without clean-slate technologies, which we discuss in §VII. The emerging CXL interconnect is bound to bridge this gap by leveraging a widely deployed high-bandwidth serial interface.

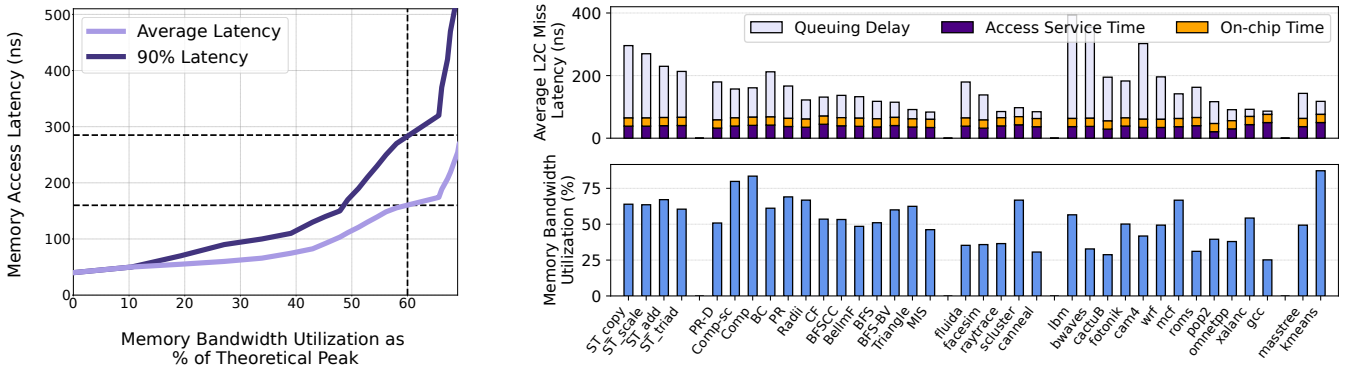
### B. The High-bandwidth CXL Interconnect

CXL [12] is a recent interconnect standard developed as a concerted industry effort for a standardized interconnect to replace a wide motley collection of competing solutions (e.g., OpenCAPI [60], Gen-Z [15], CCIX [13]). CXL aims to present a unified solution for coherent accelerators, peripheral IO devices, and memory expansion devices and is bound to become a dominant interconnect, as PCIe has been for peripheral devices over the past twenty years. We focus on CXL's capability of attaching DDR-based memory ("Type-3" devices) over PCIe to the processor and making it directly accessible via default load/store instructions. CXL's underlying PCIe physical layer affords higher bandwidth per pin at the cost of increased latency. Therefore, most recent works perceive CXL as a technology enabling an *auxiliary* slower memory tier directly attached to the processor. In contrast, we argue that despite its associated latency overhead, CXL can play a central role in future memory systems design, *replacing*, rather than simply augmenting, DDR-attached memory in bandwidth-constrained manycore server processors.

### C. Scaling the Memory Bandwidth Wall with CXL

CXL's high bandwidth owes to its underlying PCIe physical layer. PCIe [50] is a high-speed serial interface featuring multiple independent lanes capable of bi-directional communication using just four pins per lane: two for transmitting data (TX), and two for receiving data (RX). Data is sent over each lane as a serial bit stream at very high bit rates in an encoded format.

Figure 1 shows the bandwidth per pin for PCIe and DDR, derived by dividing each interface's peak bandwidth by the processor pins required: 160 for DDR and 4 per lane for




(a) Average and p90 memory access latency in a DDR5-4800 channel (38.4GB/s) at varying bandwidth utilization points. p90 grows faster than the average latency.

(b) Memory latency breakdown (DRAM access time and queuing delay) and % memory bandwidth utilization for a range of workloads. Higher utilization increases queuing delay, which constitutes 60% of L2 miss latency on average.

Fig. 2: Queuing delays dictate memory access time on a loaded system.

PCIe. The present bandwidth gap is 4× (PCIe5.0 vs. DDR5-4800) and may grow to 8× by 2025, even when ignoring that PCIe's stated bandwidth is *per direction*, while DDR's reported values refer to *combined* read/write bandwidth. For example, a DDR5-4800's channel needs 160 pins to drive an aggregate bandwidth of 38.4GB/s. Requiring only 20% of these pins, 8 PCIe5.0 lanes (over which CXL operates) offer 32GB/s per direction—i.e., a theoretical peak of 32GB/s for reads *and* 32GB/s for writes.

### D. CXL Latency Concerns

CXL's increased bandwidth comes at the cost of higher latency compared to DDR. The latency premium for the simplest CXL memory attached to a single CPU is currently about 50ns, based on reports by Intel and PLDA on existing CXL-2.0 controller implementations [51]. The expectation of such high latency premium has led memory system researchers and designers to predominantly focus on CXL as a technology for enabling a secondary tier of slower memory that augments conventional DDR-attached memory. Notably, CXL's latency overhead will likely further shrink as the technology matures.

We show in §III that a 50ns latency overhead often pales in comparison to queuing delays at the memory controller that are common in a commercially significant class of servers, and such queuing delays can be curtailed by CXL's considerable bandwidth boost. Furthermore, ample bandwidth availability can be leveraged to devise mechanisms that trade off additional bandwidth usage for latency optimization. We thus argue that CXL is a strong candidate to completely *replace* the DDR-attached memory for the subset of server processors optimized for high-throughput, memory-intensive workloads.

## III. Effective Memory Access Latency Breakdown

We now characterize the latency breakdown for a loaded memory system, highlighting the latency opportunity reduction via a bandwidth-rich CXL-based memory system. Figure 2a shows a DDR5-4800 channel's memory access latency as its load increases. We model the memory using DRAMSim3 [31] and control the load with random memory accesses of configurable arrival rate. The resulting load-latency curve is shaped by queuing effects at the memory controller.

When the system is unloaded, a hypothetical CXL interface adding 50ns to each memory access would correspond to a seemingly prohibitive 125% latency overhead compared to the unloaded latency of ∼40ns. However, as the memory load increases, latency rises exponentially, with its average value increasing by 3× and 4× at 50% and 60% load, respectively. p90 tail latency grows even faster, rising by 4.7× and 7.1× at the same load points. As a result, in a loaded system, trading off additional interface latency for considerably higher bandwidth availability can yield significant net latency gain.

To illustrate, consider a baseline DDR-based system operating at 60% utilization, corresponding to 160ns average and 285ns p90 memory access latency (Figure 2a). A CXL-based alternative offering a 4× memory bandwidth boost would shrink the system's bandwidth utilization to 15%, yielding 37%/61% lower average/p90 memory access latency compared to the baseline, *after* accounting for CXL's latency premium.

Figure 2a shows that the system experiences queuing effects at bandwidth utilization as low as 20%, that are initially reflected on tail latency. Beyond 40% utilization, queuing effects also noticeably affect average latency. Utilization beyond such level is common, as we show with our simulation of a server processor with one DDR5 memory channel per 12 cores across server and desktop applications, representing workloads that run on modern high-end manycore servers (methodological details in §V). Figure 2b shows that with all processor cores under use, the vast majority of workloads exceed 30% memory bandwidth utilization, and most exceed 50% utilization (except several workloads from SPEC and PARSEC benchmarks).

Figure 2b also breaks down the average memory access time, as measured from the L2 cache miss register, into on-chip time (NOC and LLC), DRAM service time, and queuing delay at the memory controller. We observe a trend in high bandwidth consumption leading to long queuing delays, although queuing delay doesn't present itself as a direct func-

tion of bandwidth utilization. Queuing delay is also affected by application characteristics such as read/write pattern and spatio-temporal distribution of accesses (e.g., bursts and inter-bank load imbalance). In Figure 2b's wide range of workloads, queuing delay constitutes 60% of the L2 miss latency on average, and up to 84% in the case of *lbm*. On-chip time is less pronounced but sizable, accounting for 15% of L2 miss latency on average.

In conclusion, a bandwidth-rich memory system can mitigate the queuing and on-chip components of memory access time. Higher bandwidth availability directly ameliorates queuing, but can also be leveraged to curb on-chip time by removing LLC access from the critical path. Considering the dual role of the LLC as a latency reduction mechanism and a memory traffic filter, memory bandwidth abundance diminishes the significance of the latter, motivating selective Concurrent Access of the LLC and Memory (CALM). Hence, the CXL-afforded bandwidth boost can be used to mitigate both queuing delays and on-chip time, more than compensating for the CXL interface's latency overhead for effective memory access time reduction.

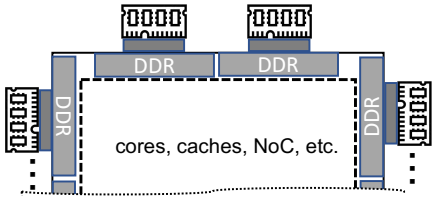
## IV. The CoaXiaL Server Architecture

Figure 3b depicts CoaXiaL, which replaces each on-chip DDR5 channel with several CXL channels, providing 2–4× higher aggregate memory bandwidth to the processor. Figure 3a shows the baseline DDR-based design for comparison. Each CXL channel is attached to a "Type-3" CXL device, which features an unmodified DDR5 controller connecting to DRAM. The processor implements the `CXL.mem` protocol of the CXL standard, which orchestrates data consistency and memory semantics management. The implementation of the caches and cores remains unchanged, as the memory controller still supplies 64B cache lines.

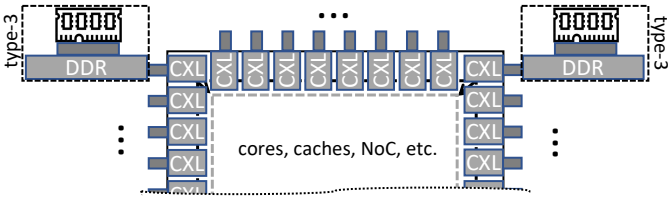
### A. Processor Pin Considerations

A DDR5-4800 channel requires at least 160 processor pins to account for data and ECC bits, command/address bus, etc. (§II-A), for a peak *unidirectional* bandwidth of 38.4GB/s. A single PCIe lane delivers 4GB/s of *bidirectional* bandwidth. and requires just four processor pins—two per TX/RX direction. Higher-bandwidth channels are constructed by grouping 2, 4, 8, 12, or 16 lanes.

We opt for an x8 (i.e., 8-lane) configuration with a peak theoretical bandwidth of 32GB/s *per direction*, requiring 32 pins—5× fewer than a DDR5 channel's. Factoring in PCIe and CXL's header overheads, an x8 channel's effective bandwidth is 26GB/s in the DRAM-to-CPU (RX) direction and 13GB/s in the opposite direction (TX) [51]. Considering a typical 2:1 Read:Write ratio, only 25.6GB/s of the DDR5-4800 channel's theoretical bandwidth would be used in the RX direction, and 13GB/s in the TX direction. Additionally, a DDR channel's effective bandwidth is typically 70–90% of its theoretical peak. Hence, an x8 CXL channel can support a full DDR5 channel on the Type-3 device without becoming a bottleneck.




(a) Baseline DDR-based server.




(b) CoaXiaL replaces each DDR channel with several CXL channels. Each CXL channel connects to a type-3 device with one DDR memory channel.

Fig. 3: Overview of the baseline and CoaXiaL systems.

### B. Silicon Area Considerations

Based on processor pin requirements alone, CoaXiaL can replace each DDR channel (i.e., PHY and memory controller) with five x8 PCIe PHY and controllers, for a 5× memory bandwidth boost. However, the relative pin requirements of DDR and PCIe do not directly correspond to their relative on-chip silicon area requirements. Lacking publicly available silicon area information, we study relevant die shots.

Table I shows the relative silicon die area key components of the processor account for. We derive the relative area of 1MB LLC, PCIe, and DDR from a Golden Cove die shot (Intel 10nm), and the size of a Zen 3 core relative to 1MB LLC from a Zen 3 die shot (TSMC 7nm) [34]. While the density of the two process technologies is comparable (reportedly within 7–11%), we only compare component areas within the same technology (*not* across technologies) to derive their relative size. An x8 PCIe controller accounts for 55% of a DDR controller's area. Hence, replacing each DDR controller *C* with four x8 PCIe controllers requires 2.2× more silicon area than *C*. However, DDR controllers account for a small fraction of the total CPU die. Notably, the relative on-chip area of DDR versus PCIe, which is the most consequential aspect of our analysis, is similar in both Intel and AMD chips [34], [58].

Leveraging Table I's data, we consider alternative CoaXiaL server designs, shown in Table II. We focus on high-core-count servers optimized for throughput, such as the AMD EPYC Bergamo (128 cores) [42], and Intel Granite Rapids (128 cores) and Sierra Forest (144 cores) [43]. All three feature 12 DDR5 channels, resulting in a core-to-memory-controller (core:MC) ratio of 10.7:1 to 12:1. A common design choice to accommodate such high core counts is a reduced LLC capacity; e.g., moving from the 96-core Genoa [56] to the 128-core Bergamo, AMD halves the LLC per core to 2MB. We thus consider a 144-core baseline server processor with 12 DDR5 channels and 2MB of LLC per core (Table II, first row).

With pin count as its only limitation, CoaXiaL-5× re-

TABLE I: Area of processor components, relative to 1MB of LLC (L3 cache).

| | |
|---|---|
| L3 cache (1MB) | 1 |
| Zen 3 Core (including 512 KB L2) | 6.5 |
| x8 PCIe (PHY + ctrl) | 5.9 |
| DDR channel (PHY + ctrl) | 10.8 |

TABLE II: DDR-based versus alternative CoaXiaL server configurations.

| Server design | Core count | LLC per core | Memory interfaces | Relative mem. BW | Relative area | Comment | |
|---|---|---|---|---|---|---|---|
| DDR-based | 144 | 2 MB | 12 DDR | 1× | 1 | baseline | |
| CoaXiaL-5× | | | 60 x8 CXL | 5× | 1.17 | iso-pin | |
| CoaXiaL-2× | | | 24 x8 CXL | 2× | 1.01 | iso-area | iso-LLC |
| CoaXiaL-4× | | 1 MB | 48 x8 CXL | 4× | | | balanced |
| CoaXiaL-asym | | | 48 x8 CXL-asym (see §IV-D) | asym. R/W | | | max BW |

places each DDR channel with 5 x8 CXL interfaces, for a 5× bandwidth increase. However, that results in a 17% die area increase to accommodate all the PCIe PHYs and controllers. Hence, we consider two iso-area alternatives. CoaXiaL-2× leverages CXL to double memory bandwidth without any microarchitectural changes. CoaXiaL-4× quadruples the available memory bandwidth compared to the baseline CPU, but halves the total LLC size from 288MB to 144MB.

### C. Latency Mitigation via Concurrent LLC/Memory Access

All CoaXiaL configurations in §IV-B leverage CXL to boost memory bandwidth availability at the cost of latency. Given the high LLC miss ratios of bandwidth-intensive workloads, we propose Concurrent Access of LLC and Memory (CALM) as a latency mitigation technique, whereby selected L2 misses look up memory and the LLC in parallel, thereby removing LLC lookup latency from the critical path. Due to its ample bandwidth, CoaXiaL is more amenable to CALM, as every concurrent LLC/Memory access decision that hits in the LLC adds memory bandwidth pressure, which is already a constrained resource in the baseline DDR-based system.

Figure 4 summarizes CALM logic. An L2 miss results in either a normal (i.e., serial LLC and memory access) or a CALM memory hierarchy access. In the case of a CALM access, the L2 receives two responses: one from the LLC and one from memory. If the requested data already resides on chip, the response from memory may be stale. To preserve coherence, CALM lookups *always* wait for a response from the LLC, even in the rare case when memory's response returns earlier, to ensure the most recent data value (from LLC, if present, otherwise from memory) is returned to the processor.

We consider two designs for producing the per-L2-miss decision of whether to perform CALM. The first design, $CALM_R$, regulates L2 misses that perform CALM to ensure the memory bandwidth utilization remains below $R$%. By monitoring L2 misses that hit or miss in the LLC, each L2 controller can estimate its memory bandwidth demand with and without the LLC acting as a filter ($bw_{filtered}$ and $bw_{unfiltered}$). If the estimated LLC-filtered memory bandwidth ($bw_{filtered}$) exceeds $R$, CALM lookup is not performed as the system is already bandwidth-constrained. Otherwise, the L2 miss performs CALM with a probability of $min(1, \frac{R-bw_{filtered}}{bw_{unfiltered}})$, thus adjusting to the estimated bandwidth utilization and $R$.

In the second design, an L2 miss performs CALM based on its predicted probability of also missing in the LLC. We employ the MAP-I to drive this decision, a PC-based predictor from prior work, shown to be highly accurate [48].

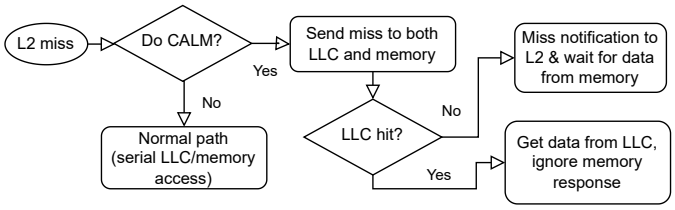



Fig. 4: Flowchart of Concurrent LLC/Memory Access.

We evaluate $CALM_R$ and MAP-I in §VI-B and find that $CALM_{70\%}$ (i.e., $CALM_R$ with $R = 70\% \times bw_{max}$) strikes the best balance of performance and simplicity. Unlike MAP-I, $CALM_R$ does not require additional hardware for the predictor's state or making the PC available at the L2. Our default CoaXiaL design therefore employs $CALM_{70\%}$.

### D. Additional Opportunities from CXL Standard Evolution

Unlike DDR, CXL has dedicated pins and wires for each data movement direction (RX and TX). The PCIe standard defines a 1:1 match of TX and RX pins: e.g., an x8 PCIe configuration implies 8 TX and 8 RX lanes. While uniform bandwidth provisioning in each direction is reasonable for a peripheral device like a NIC, memory traffic directionality is inherently imbalanced. Because (i) most workloads read more data than they write and (ii) every written cache block must typically be read first, R:W ratios are usually in the 3:1 to 2:1 range rather than 1:1. Thus, in the typical 1:1 provisioning, when read bandwidth is depleted, write bandwidth remains underutilized. Given this insight and that serial interfaces do not fundamentally require 1:1 RX:TX bandwidth provisioning [64], we consider a CoaXiaL design with asymmetric RX:TX lane provisioning to better match memory traffic characteristics. *While currently unsupported by the PCIe standard, we investigate the potential performance benefits of revisiting that statutory restriction.* We call such a channel *CXL-asym*.

We consider a system leveraging such CXL-asym channels to compose a **CoaXiaL-asym** configuration. An x8 CXL channel consists of 32 pins, 16 each way. Without the current 1:1 PCIe restriction, CoaXiaL-asym repurposes the same pin count to 20 RX pins and 12 TX pins, resulting in 40GB/s RX and 24GB/s TX of raw bandwidth. Accounting for PCIe and CXL's header overheads, the realized bandwidth is 32GB/s for reads and 10GB/s for writes (compared to 26GB/s for reads and 13GB/s for writes in a normal x8 CXL channel) [51]. While such asymmetric provisioning must be determined at processor design time, our proposed balance outperforms

the symmetric design for all of our workloads (see §VI-C). CoaXiaL-asym does not require processor modifications beyond the CXL interface.

To utilize CoaXiaL-asym's additional read bandwidth, we provision two DDR controllers per CXL-asym channel on the Type-3 device. The number of CXL channels and their on-die area overhead remain unchanged. While CXL-asym's read bandwidth can only support 62% of the 52GB/s peak combined read bandwidth of two DDR channels (assuming a 2:1 R:W ratio), queuing delays at the DDR controller typically become significant at a much lower utilization point (Figure 2a). Therefore, CoaXiaL-asym still provides sufficient bandwidth to reduce relative utilization and contention at all queues (CXL and DDR) on the way to memory.

### E. CoaXiaL *Benefits on Memory Capacity and Cost*

While this paper's scope focuses on the performance impact of a CXL-based memory system, CoaXiaL can also have noteworthy benefits on memory capacity and cost. Servers optimized for memory capacity deploy two high-density DIMMs per DDR channel (2DPC). First, such 2DPC configurations increase capacity over 1DPC at the cost of ∼15% memory bandwidth. Second, DIMM cost grows superlinearly with density; for example, 128GB/256GB DIMMs cost 5×/20× more than 64GB DIMMs. By enabling more DDR channels, CoaXiaL allows the same or higher DRAM capacity with 1DPC and lower-density (i.e., lower-cost) DIMMs.

## V. Evaluation Methodology

**System configurations.** We compare CoaXiaL to a typical DDR-based server processor by simulating the following system configurations in ChampSim [1] coupled with DRAMsim3 [31]. Table III summarizes the configuration parameters used.

- *DDR-based baseline*. We simulate 12 cores and one DDR5-4800 memory channel (comprising two 32-bit sub-channels) as a scaled-down version of Table II's 144-core CPU.
- CoaXiaL *servers*. We evaluate CoaXiaL-2×, CoaXiaL-4×, and CoaXiaL-asym (Table II). "CoaXiaL" without a qualifier implies the -4× configuration. All CoaXiaL variants employ $CALM_{70\%}$ by default, as described in §IV-C.

TABLE III: System parameters used for simulation.

| | DDR baseline | CoaXiaL-* |
|---|---|---|
| CPU | 12 OoO cores, 2.4GHz, 4-wide, 256-entry ROB | |
| L1 | 32KB L1-I & L1-D, 8-way, 64B blocks, 4-cycle hit | |
| L2 | 512 KB, 8-way, 8-cycle hit | |
| LLC | distributed, shared & non-inclusive, 16-way, 20-cycle hit | |
| | 2 MB/core | 1–2 MB/core (see Table II) |
| Memory | DDR5-4800 [40], 2 sub-channels per channel, 1 rank per sub-channel, 32 banks per rank | |
| | 1 channel | 2–4 CXL-attached channels (see Table II) 8 channels for CoaXiaL-asym (see §IV-D) |
| NoC | 2D mesh, 3 cycles/hop | |

**CXL performance modeling.** For CoaXiaL, we model CXL controllers and PCIe bus on both the processor and the Type-3 device. A CXL port incurs an unloaded uni-directional delay of 12.5ns accounting for flit-packing, encoding/decoding, packet processing, etc., as reported by PLDA and Intel's IP [47], [51]. The PCIe bus traversal latency is determined by the link direction, bus width, and channel bandwidth. As discussed in §IV-D, the 32GB/s x8 CXL channel bandwidth results in 26/13 GB/s RX/TX goodput when header overheads are factored in (32/10 GB/s RX/TX for CXL-asym). Therefore, an 8-bit CXL channel receives 64B line in 2.5ns (2ns for 10-bit CXL-asym) and transmits it in 5.5ns (9ns for CXL-asym). Overall, CoaXiaL adds a minimum of $4\times12.5ns+2.5ns=52.5ns$ for reads and 55.5ns for writes (or more, in §VI-D's sensitivity analysis) from/to DRAM. Additionally, the CXL controller maintains message queues to buffer requests, capturing any queuing effects in our evaluation.

**Workloads.** We evaluate 36 diverse workloads after fast-forwarding to their region of interest. We deploy the same workload on all cores and simulate 200 million instructions per core after 50 million instructions of warmup per core.

- *Graph analytics:* We use 13 workloads from the LIGRA benchmark suite [52].
- *STREAM:* We run the STREAM benchmark's four kernels (*copy, scale, add, triad*) [39] to represent bandwidth-intensive matrix operations in which ML workloads spend a significant portion of their execution time.
- *SPEC and PARSEC:* We evaluate 12 SPEC-speed 2017 [54] workloads in *ref* mode and five PARSEC workloads [6].
- We evaluate *masstree* [37] and *kmeans* [33] to represent key value store and data analytics workloads, respectively.

Table IV summarizes all our evaluated workloads, along with their IPC and MPKI as measured on the DDR-based baseline.

TABLE IV: Workload IPC and LLC MPKI metrics for the DDR-based baseline system.

| Application | IPC | LLC MPKI | Application | IPC | LLC MPKI |
|---|---|---|---|---|---|
| **SPEC** | | | **Ligra** | | |
| lbm | 0.14 | 64 | PageRank Delta | 0.30 | 27 |
| bwaves | 0.33 | 14 | Comp.-shortcut | 0.34 | 48 |
| cactusBSSN | 0.68 | 8 | Components | 0.36 | 48 |
| fotonik3d | 0.32 | 22 | BC | 0.33 | 34 |
| cam4 | 0.87 | 6 | PageRank | 0.36 | 40 |
| wrf | 0.61 | 11 | Radii | 0.41 | 33 |
| mcf | 0.79 | 13 | CF | 0.8 | 12 |
| roms | 0.77 | 6 | BFSCC | 0.65 | 17 |
| pop2 | 1.5 | 3 | BellmanFord | 0.82 | 9 |
| omnetpp | 0.50 | 10 | BFS | 0.66 | 15 |
| xalancbmk | 0.50 | 12 | BFS-Bitvector | 0.84 | 15 |
| gcc | 0.27 | 19 | Triangle | 0.61 | 21 |
| **STREAM** | | | MIS | 1.19 | 8 |
| Stream-copy | 0.17 | 58 | **PARSEC** | | |
| Stream-scale | 0.21 | 48 | fluidanimate | 0.73 | 7 |
| Stream-add | 0.16 | 69 | facesim | 0.74 | 6 |
| Stream-triad | 0.18 | 59 | raytrace | 1.1 | 5 |
| **KVS & Data Analytics** | | | streamcluster | 0.95 | 14 |
| Masstree | 0.37 | 21 | canneal | 0.61 | 7 |
| Kmeans | 0.50 | 36 | | | |

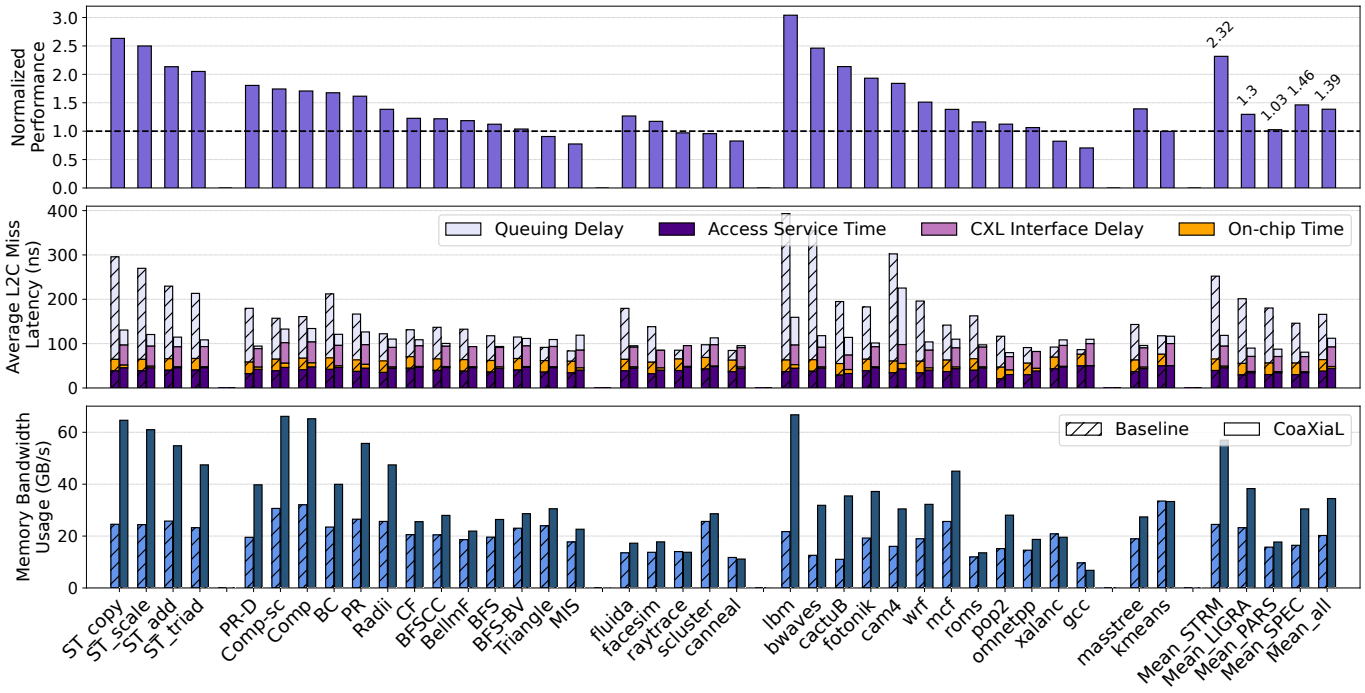



Fig. 5: CoaXiaL speedup over DDR-based baseline (top), memory access latency breakdown (middle), and memory bandwidth usage (bottom). Despite higher absolute bandwidth usage, CoaXiaL operates at lower utilization, due to its 4× higher bandwidth availability. Reduced queuing lowers average memory access latency, resulting in 1.39× average speedup.

**Figures of merit.** Our key metric is the speedup (normalized IPC improvement compared to the baseline) afforded by CoaXiaL configurations. We also analyze the impact on memory access latency and bandwidth utilization.

## VI. Evaluation Results

In this section, we perform a thorough evaluation of CoaXiaL and demonstrate the following key takeaways:

1) CoaXiaL drastically reduces queuing delays and on-chip time, resulting in 29% lower average L2 miss latency and 1.39× average speedup (§VI-A).
2) Bandwidth-rich systems like CoaXiaL are amenable to concurrent LLC/memory access techniques that can effectively curb on-chip time for sizable latency reduction (§VI-B).
3) Provisioning CXL lanes in a read/write demand-aware manner considerably improves performance (by 13%) compared to the default 1:1 read:write ratio (§VI-C).
4) Even with an increased 70ns CXL latency overhead, CoaXiaL achieves a substantial 1.26× average speedup (§VI-D).
5) Even at 66% server utilization—or 8:1 core:MC ratio—CoaXiaL delivers a 1.17× speedup (§VI-E).
6) In addition to major performance gains, CoaXiaL affords 25% lower *EDP* and 47% lower $ED^2P$ (§VI-F).

### A. Main Results

Figure 5 (top) shows the performance of CoaXiaL-4× relative to the baseline DDR-based system. Most workloads exhibit significant speedup, up to 3× for *lbm* and 1.39× on average. The speedup for 15 of the 36 workloads exceeds 1.5×. Seven workloads lose performance, with gcc most significantly impacted at 26% IPC loss. Workloads that suffer performance loss are those with low to moderate memory traffic, heavy dependencies among memory accesses (low memory-level parallelism), and high LLC hit rate.

Figure 5 (bottom) shows memory bandwidth usage for both systems. CoaXiaL distributes memory requests over more channels which, given its 4× higher bandwidth availability versus the baseline, reduces relative bandwidth utilization, and thus contention and queuing delay for memory-intensive workloads. Figure 5 (middle) demonstrates this reduction with a breakdown of the average L2 miss latency into on-chip time (NoC and LLC), DRAM access service time, queuing delay, and CXL interface delay (only applicable to CoaXiaL).

CoaXiaL enables several workloads to use significantly more memory bandwidth. Despite the absolute bandwidth use increase, the 4× higher bandwidth availability results in an average bandwidth utilization drop from 54% to 34%, reducing queuing delays. For instance, *stream-copy* is bottlenecked by the baseline's constrained bandwidth, resulting in an average queuing delay over 230ns that dominates the overall L2 miss latency. For the same workload, CoaXiaL reduces queuing delay to just 34ns, thanks to its ample bandwidth, and also average on-chip time from 24ns to 6ns, thanks to $CALM_{70\%}$. These two effects combined more than compensate for the 45ns CXL interface latency overhead an *average* L2 miss

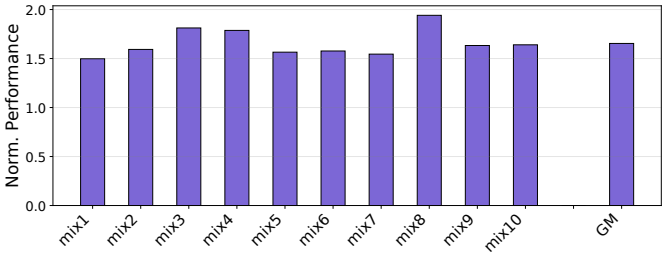
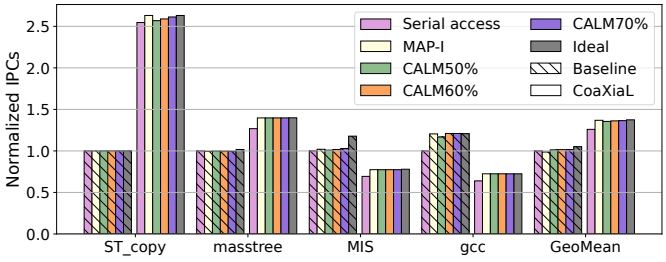
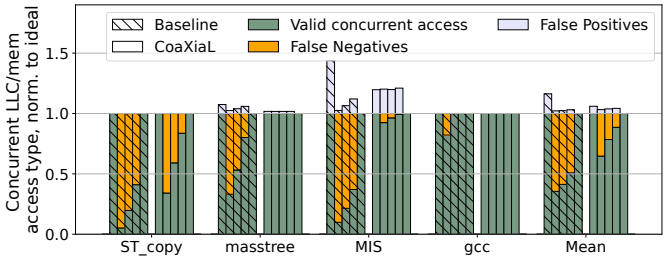



Fig. 6: CoaXiaL's performance for workload mixes, normalized to DDR-based server.




(a) Normalized performance with different CALM mechanisms.




(b) Characterization of CALM decisions across different CALM mechanisms. Each of the five bars in each cluster corresponds to a CALM mechanism, in the following order: MAP-I, $CALM_{50\%}$, $CALM_{60\%}$, $CALM_{70\%}$, Ideal.

Fig. 7: Sensitivity study of different CALM mechanisms.

experiences (50ns $\times$ LLC miss-ratio), reducing *stream-copy*'s baseline L2 miss latency from 336ns to just 176ns. Lower effective memory access latency enables CoaXiaL to drive memory requests at a 2.6$\times$ higher rate and achieve commensurate speedup.

For most workloads, CoaXiaL yields lower L2 miss latency than the baseline, despite the CXL interface's latency overhead. It is noteworthy that speedup, while generally correlating well with latency reduction, is not a direct function of memory access latency. The resulting speedup is also affected by each workload's specific characteristics (e.g., memory-level parallelism, temporal memory access pattern, etc.). Therefore, the same latency reduction across two different workloads does not necessarily yield the same speedup.

On average, workloads experience 102ns in queuing delay and 24ns in on-chip time on top of $\sim$40ns DRAM service time. CoaXiaL slashes queuing delay by 5$\times$, to just 20ns, and on-chip time by 66% to 8ns, on average. Overall, CoaXiaL reduces average memory access latency by 29%.

**Mixed Workloads.** In Figure 5's results, all cores execute the same workload. We also evaluated 10 workload mixes, shown in Figure 6, with each mix comprising 12 randomly sampled workloads from Table IV. CoaXiaL's min/max/geomean speedup is $1.5\times/1.9\times/1.7\times$. Thus, our main results likely *underestimate* the extent of performance gain from CoaXiaL. Mixed workloads are common in throughput-oriented servers, and are more likely to contain bandwidth-intensive workloads running with latency-sensitive workloads, driving up the average memory utilization in the baseline.

### B. Effect of Concurrent Access of LLC and Memory (CALM)

CoaXiaL's performance gains are a composite effect of its bandwidth superiority and its $CALM_{70\%}$ mechanism that removes the LLC from the critical path. Figure 7 decomposes the gains delivered by $CALM_{70\%}$, compares against alternative CALM mechanisms, and shows the effect of adding such a mechanism to the baseline as well. We evaluate both systems (i) with the default serial LLC/memory access; (ii) combined with §IV-C's different CALM mechanisms; and (iii) with an ideal predictor that always correctly predicts if an L2 miss also misses in the LLC. For brevity, we only show results for four workloads and the average behavior across all 36 workloads.

Figure 7a shows speedup relative to baseline with serial LLC/memory access. For workloads where CoaXiaL wins considerably, removing the LLC from the critical path doesn't substantially benefit the baseline, as bandwidth scarcity is the main bottleneck. All versions of CALM can considerably benefit the baseline with latency-bound workloads like gcc. However, the baseline's *average* gain from CALM is negligible, due to its bandwidth-constrained nature, while all CALM mechanisms substantially benefit CoaXiaL. Due to the high LLC miss ratio of 88% in CoaXiaL, all predictors perform, on average, very close to an ideal predictor. Hence, we use $CALM_R$ over MAP-I due to its simplicity and choose $R = 70\%$ as it marginally outperforms lower thresholds. $CALM_{70\%}$ boosts CoaXiaL's speedup over baseline from 1.28$\times$ (where LLC and memory access are serialized) to 1.39$\times$.

Figure 7b further analyzes the CALM mechanisms' behavior. The penalty of a false positive (i.e., an assumed LLC miss hits) is wasteful memory bandwidth use, while the penalty of a false negative (i.e., an assumed LLC hit misses) is increased latency, as LLC and memory access get serialized. Due to CoaXiaL's ample bandwidth, false positives are preferable to false negatives. The opposite is true for the bandwidth-bottlenecked baseline. $CALM_{70\%}$ is the best choice for CoaXiaL, as it minimizes false negatives. On average, $CALM_{70\%}$ incurs false positives corresponding to 4% of memory accesses, while false negatives make up 11% of all LLC misses. MIS is an outlier, where false positives incur a 21% increase in memory accesses. Despite its high value, that memory bandwidth overutilization is not detrimental to performance as the total memory bandwidth utilization remains below 15% of CoaXiaL's theoretical peak

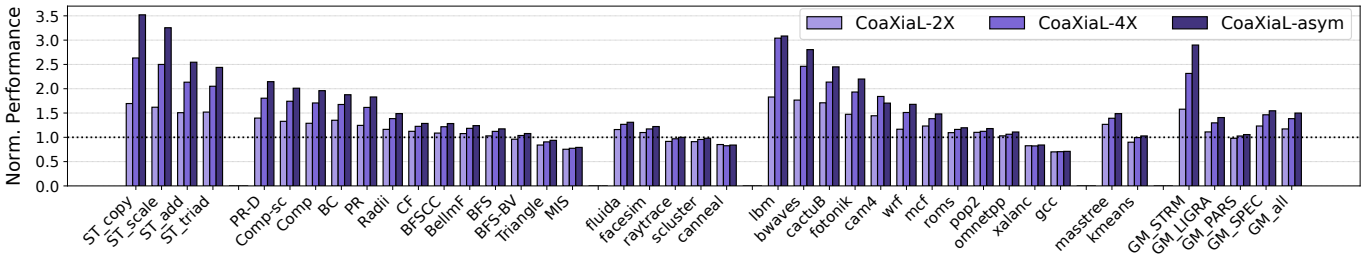



Fig. 8: Performance of different CoaXiaL configurations normalized to DDR-based server baseline. CoaXiaL-4× considerably outperforms CoaXiaL-2×, despite half the LLC size. CoaXiaL-asym provides a 13% speedup over CoaXiaL-4×.

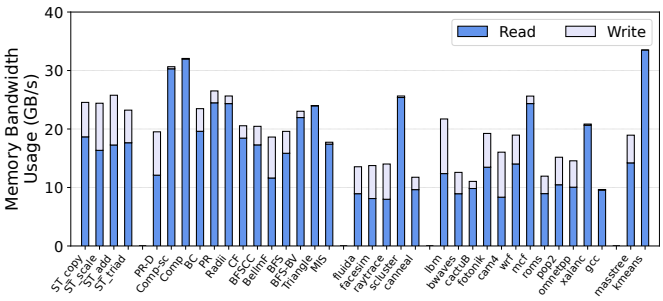



Fig. 9: Read/write bandwidth usage in the baseline system.

(see Figure 5 (bottom)). Due to the high LLC miss ratio, false negatives far exceed false positives, suggesting that CALM mechanisms are better suited to bandwidth-rich systems like CoaXiaL.

### C. Alternative CoaXiaL Designs

Figure 8 evaluates the two alternative CoaXiaL designs from §IV—CoaXiaL-2× and CoaXiaL-asym—in addition to our default CoaXiaL-4×. CoaXiaL-2× achieves a 1.17× average speedup over the baseline, down from the 1.39× gain of CoaXiaL-4×, confirming that doubling memory bandwidth availability at the cost of halving the LLC is beneficial for practically all workloads. CoaXiaL-asym boosts performance by 1.52× on average—a considerable 13% gain over CoaXiaL-4×—and no workload is negatively affected by CoaXiaL-asym's reduced write bandwidth. This result highlights the opportunities of tailoring bandwidth provisioning to the workloads' asymmetric read/write demands.

To further explain CoaXiaL-asym's performance gains, Figure 9 breaks down the baseline system's bandwidth utilization into read and write traffic. The results align with the expectation that read traffic significantly outweighs write traffic: the average R:W ratio is 3.7:1 across our 36 workloads. Cam4 is the most write-intensive workload, approaching a 1:1 R:W ratio. However, even such workloads benefit from CoaXiaL-asym's provisioned 3.2:1 R:W bandwidth, because (i) the reduced available write bandwidth of 40GB/s remains sufficient; and (ii) overprovisioning read bandwidth optimizes for performance-critical loads, while writebacks are more tolerant to queuing delays.

### D. Sensitivity to CXL's Latency Overhead

We based our evaluation so far on an unloaded latency penalty of 50ns for CXL, based on expectations set by current technological evidence (see §II-D and §V). We also evaluate a more pessimistic latency overhead of 70ns, as early products may incur higher latency. Such higher latency may also better represent CXL-attached memory located further from the CPU, or devices with an additional multiplexing overhead (e.g., memory devices shared by several processors [21], [30]).

Figure 10 shows CoaXiaL's performance at 50ns and 70ns CXL interface latency overhead, normalized to the DDR-based baseline. Although a 70ns overhead reduces CoaXiaL's average speedup, it remains significant at 1.26×. Several memory-intensive workloads continue to enjoy drastic speedups of over 50%, but more workloads (ten, up from seven in the case of 50ns latency penalty) take a performance hit. These results imply that while a CoaXiaL design with a higher CXL latency is still worth pursuing, it should be used selectively for memory-intensive workloads. Deploying different server classes for different optimization goals is common practice not only in both public [20] and private clouds (e.g., different web and backend server configurations) [16], [23].

### E. Sensitivity to Core Utilization

Figure 11 evaluates CoaXiaL under varying levels of system utilization by provisioning proportionately less work on a fraction of the system's cores. We first study the extreme case of using a single core on our 12-core simulated system (8% utilization). In this scenario, the vast majority of workloads suffer performance degradation with CoaXiaL, for a 27% average slowdown. The extreme single-core experiment showcases a worst-case scenario that every operator strives to avoid, as the whole system is severely underutilized.

We then increase the system utilization to 33% and 66%, by deploying workload instances on 4 and 8 cores of the 12-core CPU, respectively. CoaXiaL's bandwidth abundance gradually starts paying off, by eliminating the slowdown at 33% utilization for most workloads, and then delivering significant gains—1.17× on average and up to 2.67×—even at 66% utilization. The 66% utilization point is also a good proxy for a fully loaded system where cores and DDR controllers are provisioned at an 8:1 ratio. An 8:1 core:MC ratio is the design point of many server processors with fewer than 100 cores

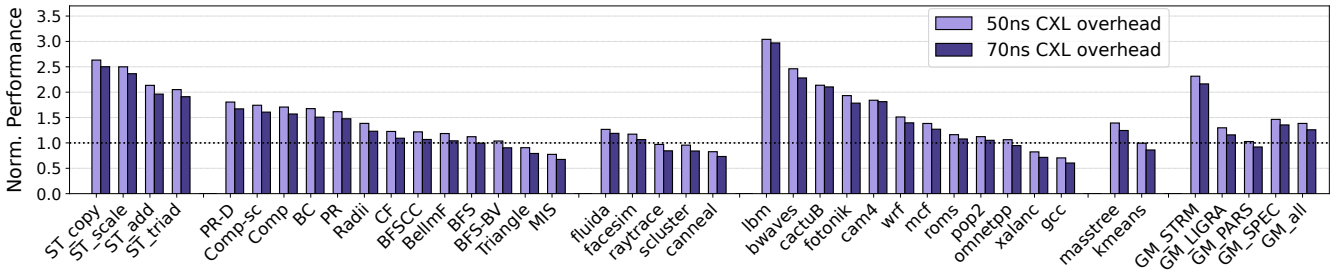



Fig. 10: CoaXiaL's performance for different CXL latency premium, normalized to the DDR-based server.

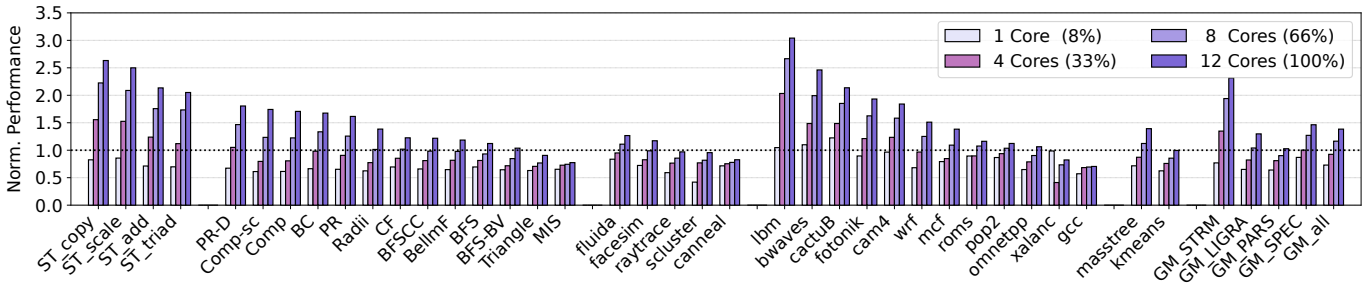



Fig. 11: Performance of CoaXiaL as a function of active cores, normalized to the DDR-based server baseline at the same number of active cores.

today, such as AMD EPYC Milan and Genoa [11], [56]. Thus, the 66% utilization results imply that CoaXiaL's approach is applicable beyond high-end throughput-oriented processors that already exhibit 12:1 core:MC oversubscription.

A few outlier workloads (cactusBSSN, bwaves, lbm) benefit from CoaXiaL's bandwidth boost even at single-core system utilization. We observe surprisingly high LLC miss latencies in the baseline system, 30–210ns on average, much higher than the average of 63ns observed across workloads. We hypothesize that these workloads exhibit highly bursty memory access behavior, leading to queuing effects that CoaXiaL ameliorates. Finally, xalanc's single-instance dataset is entirely LLC-resident, resulting in equal performance of the baseline and CoaXiaL. With 4+ instances, xalanc's working set exceeds the LLC capacity, but its memory bandwidth requirements remain too low (20% utilization even with all 12 cores active—see Figure 5 (bottom)) to benefit from CoaXiaL.

### *F. Power Requirements and Energy Efficiency*

CoaXiaL significantly improves performance while drawing more power, due to additional DDR and CXL channels. Performance and energy are two metrics at odds, as optimizing for one typically hurts the other. Servers are performance-optimized systems that routinely employ performance optimizations at the cost of non-linear increase in energy, essentially optimizing for the composite metric of *Energy Delay Product* ($EDP$ = *system power* $\times$ $CPI^2$, lower is better). For high-end servers, it is also common to use $ED^2P$ to further emphasize performance [7], [17], [53]. We evaluate CoaXiaL's impact on both metrics of $EDP$ and $ED^2P$ and additionally report performance per watt, emphasizing that this metric is more relevant for low-power than performance-optimized systems, and hence has limited value as a point of comparison. Still, we show that CoaXiaL achieves 96% of the baseline's performance-per-watt, while improving performance by 1.39$\times$ on average.

We model power for a manycore CPU similar to Intel's Sierra Forest (144 cores) [43], which has a 500W TDP, in line with other comparable CPUs (e.g., 96-core AMD EPYC Genoa [56] has a TDP of 360W). We estimate controller and interface power per DDR5 channel to be 1.1W [62], or 13W for 12 channels. CoaXiaL eschews 50% of LLC capacity to accommodate the added serial links, which reduces LLC power draw (leakage + access) from 94W in baseline to 51W (from Cacti 7.0 [4] at 22nm technology). We estimate power consumption of common components (cores, L1, L2, etc.) to be 393W (= 500 - 13 - 94). Finally, PCIe 5.0's interface power is $\sim$0.2W per lane [5], adding 77W to package power for the 384 lanes required for CoaXiaL's 48 DDR5 channels.

We model memory power using DRAMSim3's power model with a 32GB DDR5-4800MT/s RDIMM [41] per DDR5 channel. For an iso-capacity comparison, the baseline must feature 4$\times$ denser DIMMs than CoaXiaL, meaning the baseline would require 128GB RDIMMs. As power parameters for 128GB RDIMMs are not yet available, we assume the same power characteristics as 32GB RDIMMs, underestimating the baseline's power consumption. Nonetheless, even with 4$\times$ more DIMMs, CoaXiaL's DRAM power is only 2.45$\times$ higher, due to lower memory utilization than the baseline.

Table V summarizes the key power components for the baseline and CoaXiaL systems. The overall system power

TABLE V: Energy/power comparison for 144-core server.

| EDP Component | Baseline | CoaXiaL |
|---|---|---|
| Processor Core + L1 + L2 Power | 393W | 393W |
| DDR5 MC & PHY power (all) | 13W | 52W |
| LLC Power (leakage and access) | 94W | 51W |
| CXL's Interface power (idle and dynamic) | N/A | 77W |
| DDR5 DIMM power (static and access) | 146W | 358W |
| Total system power | 646W | 931W |
| Average CPI (all workloads) | 2.05 | 1.48 |
| Relative Perf/Watt (all workloads) | 1 | 0.96 |
| **EDP (all workloads)** (lower is better) | **2,715** | **2,039 (0.75×)** |
| **$ED^2P$ (all workloads)** (lower is better) | **5,566** | **3,018 (0.53×)** |

consumption is 646W for the baseline system and 931W for CoaXiaL. Thanks to CoaXiaL's 39% performance boost, the relevant server-centric metrics of EDP and $ED^2P$ drastically reduce by 25% and 47%, respectively.

## VII. Related Work

**Emerging CXL-based memory systems.** Industry is rapidly adopting CXL and already investigating its deployment in production systems to reap the benefits of memory expansion and memory pooling. Microsoft's Pond leverages CXL to pool memory across servers, improving utilization and thus cost [30]. StarNUMA employs a CXL pool to host hot data shared by many sockets and thus reduce expensive multi-hop interconnect link traversals in large NUMA machines [9]. Gouk et al. [21] prototype a practical CXL-based instance of disaggregated memory [32]. Meta leverages CXL as a memory expansion technique enabling a secondary high-capacity memory tier [38]. Ahn et al. evaluate database workloads on a hybrid DDR/CXL memory system and demonstrate that CXL-based memory expansion can be cost-efficient and performant [3]. Instead of using CXL-attached memory as a memory system expansion, CoaXiaL is the first work to investigate the potential of CXL-based memory as a *complete replacement* of DDR-attached memory for manycore server processors handling memory-intensive workloads.

**Serial interfaces.** Several prior memory system proposals leverage serial links for high-bandwidth, energy-efficient data transfers. Micron's HMC was connected to the host over 16 SerDes lanes, delivering up to 160GB/s [46]. Open Memory Interface (OMI) is IBM's high-bandwidth memory interconnect leveraging serial links to connect DDR memory [10] and is now part of the CXL Consortium. IBM's Centaur is a memory capacity expansion solution, where the host uses OMI to connect to a buffer-on-board populated with several DDR channels [59]. CoaXiaL distinctly differs from Centaur-like solutions, focusing on memory bandwidth—rather than capacity—expansion. Additionally, given OMI's very low latency overhead of 10ns, latency optimizations have not been considered in IBM's products. We find that if CXL approaches similarly low latency overhead in the future, CoaXiaL would gain massively, with 1.71× average speedup (and up to 3.1×). Moreover, combined with CALM (§IV-C), no workload would experience a performance hit, as a 10ns latency premium can be fully offset by removing the LLC from the critical path.

FBDIMM [19] leverages a similar concept to Centaur's buffer-on-board to increase memory bandwidth and capacity. An "advanced memory buffer" is connected to the processor over serial links, delivering pin abundance that enables multiple parallel interfaces to DRAM modules. Similar to CXL-attached memory, FBDIMM's drawback is increased latency.

Several proposed memory system architectures leverage high-bandwidth serial interfaces. In MeSSOS, high-bandwidth serial links connect to a DRAM cache, which is then chained to planar DRAM over DDR [63]. Ham et al. propose "disintegrated" memory controllers attached over SerDes to make the memory system more modular and support heterogeneous memory technologies [22]. Alloy combines parallel and serial interfaces to access memory, maintaining the former for lower-latency memory access [64]. Alloy's approach is closer to the hybrid DDR/CXL memory systems most ongoing CXL-related research envisions than our proposal of fully replacing DDR processor interfaces with CXL for memory-intensive servers.

**Circuit-level techniques to boost memory bandwidth.** HBM [28] and die-stacked DRAM caches offer an order of magnitude higher bandwidth than planar DRAM, but suffer from limited capacity [27], [35], [48]. BOOM [65] buffers outputs from multiple LPDDR ranks to reduce power and sustain server-level performance, but offers modest gains due to low-frequency LPDDR and limited bandwidth improvement. Chen et al. [8] dynamically reallocate power pins to boost data transfer capability from memory during memory-intensive phases, during which processors are memory bound and hence draw less power. Pal et al. [45] propose packageless processors to mitigate pin limitations and boost the memory bandwidth that can be routed to the processor. Unlike these exotic technologies, we focus on conventional processors, packaging, and commodity DRAM, aiming to reshape the memory system of server processors by leveraging the widely adopted up-and-coming CXL interconnect.

**Other memory system optimizations.** Transparent memory compression techniques can increase effective memory bandwidth [66]. Malladi et al. [36] leverage mobile LPDDR devices to design a more energy-efficient memory system for servers without performance loss. These works are orthogonal to CoaXiaL. Storage-class memory, like Phase-Change Memory [18] or Intel's Optane [24], has attracted significant interest as a way to boost a server's memory capacity, triggering research activity on transforming the memory hierarchy to best accommodate such new memories [2], [14], [29], [61]. Unlike our work, such systems often trade off bandwidth for capacity.

## VIII. Conclusion

Technological trends motivate a server processor design where all memory is attached to the processor over the emerging CXL interface instead of DDR. CXL's superior bandwidth per pin helps bandwidth-hungry manycore server processors scale the bandwidth wall. By distributing memory requests

over 4× more memory channels, CXL reduces detrimental queuing effects on the memory bus. In addition, CoaXiaL leverages ample bandwidth availability to curb on-chip latencies, via selective concurrent access of the LLC and memory. By mitigating queuing and on-chip delays, CoaXiaL more than offsets the interface latency overhead introduced by CXL. Our evaluation using a diverse range of workloads shows that CoaXiaL delivers 1.39× speedup on average, and up to 3×.

## Acknowledgments

We thank the anonymous reviewers and Marina Vemmou, Hamed Seyedroudbari, Divya Kadiyala for their feedback and suggestions that helped improve this paper. This research was supported by NSF grant 2333049, an Intel research gift, a CRNCH PhD Fellowship, and a generous donation of Azure credits from Microsoft made available via GT Cloud Hub.

## Appendix

*Note about the paper's artifact:* The following artifact had to be finalized before the paper's camera-ready version and Figures 6 and 9 were added after the artifact's submission. The data presented in these new figures can be acquired using the same, unmodified artifact. However, a side-effect of the addition is that figure numbers referenced in the artifact are inconsistent: mentions to Figures 6 and 7 in the artifact refer to the paper's Figures 7 and 8; mentions to Figures 8 and 9 in the artifact refer to the paper's Figures 10 and 11.

## References

[1] "ChampSim." [Online]. Available: https://github.com/ChampSim/ChampSim

[2] N. Agarwal and T. F. Wenisch, "Thermostat: Application-transparent Page Management for Two-tiered Main Memory," in *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XXII)*, 2017, pp. 631–644.

[3] M. Ahn, A. Chang, D. Lee, J. Gim, J. Kim, J. Jung, O. Rebholz, V. Pham, K. T. Malladi, and Y.-S. Ki, "Enabling CXL Memory Expansion for In-Memory Database Management Systems," in *Proceedings of the 18th International Workshop on Data Management on New Hardware (DaMoN)*, 2022, pp. 8:1–8:5.

[4] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "Cacti 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 14, no. 2, pp. 1–25, 2017.

[5] M. Bichan, C. Ting, B. Zand, J. Wang, R. Shulyzki, J. Guthrie, K. Tyshchenko, J. Zhao, A. Parsafar, E. Liu, A. Vatankhahghadim, S. Sharifian, A. Tyshchenko, M. D. Vita, S. Rubab, S. Iyer, F. Spagna, and N. Dolev, "A 32Gb/s NRZ 37dB SerDes in 10nm CMOS to Support PCI Express Gen 5 Protocol," in *Proceedings of the 2020 IEEE Custom Integrated Circuits Conference*, 2020, pp. 1–4.

[6] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architecture and Compilation Techniques (PACT)*, 2008, pp. 72–81.

[7] D. M. Brooks, P. Bose, S. Schuster, H. M. Jacobson, P. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. V. Zyuban, M. Gupta, and P. W. Cook, "Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors," *IEEE Micro*, vol. 20, no. 6, pp. 26–44, 2000.

[8] S. Chen, Y. Hu, Y. Zhang, L. Peng, J. Ardonne, S. Irving, and A. Srivastava, "Increasing off-chip bandwidth in multi-core processors with switchable pins," in *Proceedings of the 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 385–396.

[9] A. Cho and A. Daglis, "StarNUMA: Mitigating NUMA Challenges with Memory Pooling," in *Proceedings of the 57th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2024.

[10] T. M. Coughlin and J. Handy, "Higher Performance and Capacity with OMI Near Memory," in *Proceedings of the 2021 Annual Symposium on High-Performance Interconnects*, 2021, pp. 68–71.

[11] D. I. Cutress and A. Frumusanu, "Amd 3rd gen epyc milan review: A peak vs per core performance balance," 2021. [Online]. Available: https://www.anandtech.com/show/16529/amd-epyc-milan-review

[12] CXL Consortium, "Compute Express Link (CXL) Specification, Revision 3.0, Version 1.0," 2022. [Online]. Available: https://www.computeexpresslink.org/_files/ugd/0c1418_1798ce97c1e6438fba818d760905e43a.pdf

[13] CXL Consortium, "Compute Express Link Consortium, Inc. and CCIX Consortium, Inc. announce agreement for CXL Consortium to receive CCIX Consortium Specifications and other CCIX Consortium assets," 2023. [Online]. Available: https://www.computeexpresslink.org/post/compute-express-link-consortium-inc-and-ccix-consortium-inc-announce-agreement-for-cxl

[14] S. Dulloor, A. Roy, Z. Zhao, N. Sundaram, N. Satish, R. Sankaran, J. Jackson, and K. Schwan, "Data tiering in heterogeneous memory systems," in *Proceedings of the 2016 EuroSys Conference*, 2016, pp. 15:1–15:16.

[15] EE Times, "CXL will absorb Gen-Z," 2021. [Online]. Available: https://www.eetimes.com/cxl-will-absorb-gen-z/

[16] Engineering at Meta, " Introducing "Yosemite": the first open source modular chassis for high-powered microservers," 2015. [Online]. Available: https://engineering.fb.com/2015/03/10/core-data/introducing-yosemite-the-first-open-source-modular-chassis-for-high-powered-microservers/

[17] M. J. Flynn, P. Hung, and K. W. Rudd, "Deep submicron microprocessor design issues," *IEEE Micro*, vol. 19, no. 4, pp. 11–22, 1999.

[18] S. W. Fong, C. M. Neumann, and H.-S. P. Wong, "Phase-change memory—towards a storage-class memory," *IEEE Transactions on Electron Devices*, vol. 64, no. 11, pp. 4374–4385, 2017.

[19] B. Ganesh, A. Jaleel, D. Wang, and B. L. Jacob, "Fully-Buffered DIMM Memory Architectures: Understanding Mechanisms, Overheads and Scaling," in *Proceedings of the 13th IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 2007, pp. 109–120.

[20] Google Cloud, "Machine families resource and comparison guide." [Online]. Available: https://cloud.google.com/compute/docs/machine-resource

[21] D. Gouk, S. Lee, M. Kwon, and M. Jung, "Direct Access, High-Performance Memory Disaggregation with DirectCXL," in *Proceedings of the 2022 USENIX Annual Technical Conference (ATC)*, 2022, pp. 287–294.

[22] T. J. Ham, B. K. Chelepalli, N. Xue, and B. C. Lee, "Disintegrated control for energy-efficient and heterogeneous memory systems," in *Proceedings of the 19th IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 2013, pp. 424–435.

[23] K. M. Hazelwood, S. Bird, D. M. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, and X. Wang, "Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective," in *Proceedings of the 24th IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 2018, pp. 620–629.

[24] Intel Corporation, "Intel Optane DC Persistent Memory." [Online]. Available: https://www.intel.com/content/www/us/en/products/memory-storage/optane-dc-persistent-memory.html

[25] J. Jaffari, A. Ansari, and R. Beraha, "Systems and methods for a hybrid parallel-serial memory access," 2015, US Patent 9747038B2.

[26] JEDEC, "DDR5 SDRAM standard (JESD79-5B)," 2022.

[27] D. Jevdjic, S. Volos, and B. Falsafi, "Die-stacked DRAM caches for servers: hit ratio, latency, or bandwidth? have it all with footprint cache," in *Proceedings of the 40th International Symposium on Computer Architecture (ISCA)*, 2013, pp. 404–415.

[28] J. Kim and Y. Kim, "HBM: Memory solution for bandwidth-hungry processors," in *Hot Chips Symposium*, 2014, pp. 1–24.

[29] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *Proceedings of the 36th International Symposium on Computer Architecture (ISCA)*, 2009, pp. 2–13.

[30] H. Li, D. S. Berger, S. Novakovic, L. Hsu, D. Ernst, P. Zardoshti, M. Shah, S. Rajadnya, S. Lee, I. Agarwal, M. D. Hill, M. Fontoura, and R. Bianchini, "Pond: CXL-Based Memory Pooling Systems for Cloud Platforms," *Proceedings of the 28th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XXVIII)*, 2023.

[31] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. L. Jacob, "DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator," *IEEE Comput. Archit. Lett.*, vol. 19, no. 2, pp. 110–113, 2020.

[32] K. T. Lim, J. Chang, T. N. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated memory for expansion and sharing in blade servers," in *Proceedings of the 36th International Symposium on Computer Architecture (ISCA)*, 2009, pp. 267–278.

[33] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–136, 1982.

[34] Locuza, "Die walkthrough: Alder Lake-S/P and a touch of Zen 3," 2022. [Online]. Available: https://locuza.substack.com/p/die-walkthrough-alder-lake-sp-and

[35] G. H. Loh and M. D. Hill, "Efficiently enabling conventional block sizes for very large die-stacked DRAM caches," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2011, pp. 454–464.

[36] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile DRAM," in *Proceedings of the 39th International Symposium on Computer Architecture (ISCA)*, 2012, pp. 37–48.

[37] Y. Mao, E. Kohler, and R. T. Morris, "Cache craftiness for fast multicore key-value storage," in *Proceedings of the 2012 EuroSys Conference*, 2012, pp. 183–196.

[38] H. A. Maruf, H. Wang, A. Dhanotia, J. Weiner, N. Agarwal, P. Bhattacharya, C. Petersen, M. Chowdhury, S. O. Kanaujia, and P. Chauhan, "TPP: Transparent Page Placement for CXL-Enabled Tiered Memory," *Proceedings of the 28th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XXVIII)*, 2023.

[39] J. D. McCalpin, "Memory Bandwidth and Machine Balance in Current High Performance Computers," *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, 1995.

[40] Micron Technology Inc., "DDR5 SDRAM Datasheet," 2022. [Online]. Available: https://media-www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr5/ddr5_sdram_core.pdf

[41] Micron Technology Inc., "DDR5 32GB RDIMM with 16Gb die Datasheet," 2023. [Online]. Available: https://media-www.micron.com/-/media/client/global/documents/products/data-sheet/modules/rdimm/ddr5/mtc20f1045s1rc_srx4_rdimm_dierevg.pdf?rev=75f439eb3916453cb6032efec835ed24

[42] H. Mujtaba, "AMD EPYC Bergamo 'Zen 4C' CPUs Being Deployed In 1H 2023 To Tackle Arm CPUs, Instinct MI300 APU Back In Labs," 2022. [Online]. Available: https://wccftech.com/amd-epyc-bergamo-zen-4c-cpus-being-deployed-in-1h-2023-tackle-arm-instinct-mi300-apu-back-in-labs/amp/

[43] H. Mujtaba, "Intel Granite Rapids & Sierra Forest Xeon CPU Detailed In Avenue City Platform Leak: Up To 500W TDP & 12-Channel DDR5," 2023. [Online]. Available: https://wccftech.com/intel-granite-rapids-sierra-forest-xeon-cpu-detailed-in-avenue-city-platform-leak-up-to-500w-tdp-12-channel-ddr5/

[44] B. Nitin, W. Randy, I. Shinichiro, F. Eiji, R. Shibata, S. Yumiko, and O. Megumi, "DDR5 design challenges," in *2018 IEEE 22nd Workshop on Signal and Power Integrity (SPI)*, 2018, pp. 1–4.

[45] S. Pal, D. Petrisko, A. A. Bajwa, P. Gupta, S. S. Iyer, and R. Kumar, "A Case for Packageless Processors," in *Proceedings of the 24th IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 2018, pp. 466–479.

[46] J. T. Pawlowski, "Hybrid memory cube (HMC)," in *Hot Chips Symposium*, 2011, pp. 1–24.

[47] PLDA and AnalogX, "PLDA and AnalogX Announce Market-leading CXL 2.0 Solution featuring Ultra-low Latency and Power," 2021. [Online]. Available: https://www.businesswire.com/news/home/20210602005484/en/PLDA-and-AnalogX-Announce-Market-leading-CXL-2.0-Solution-featuring-Ultra-low-Latency-and-Power

[48] M. K. Qureshi and G. H. Loh, "Fundamental Latency Trade-off in Architecting DRAM Caches: Outperforming Impractical SRAM-Tags with a Simple and Practical Design," in *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2012, pp. 235–246.

[49] R. Rooney and N. Koyle, "Micron® DDR5 SDRAM: New Features," *Micron Technology Inc., Tech. Rep*, 2019.

[50] D. D. Sharma, "PCI Express® 6.0 Specification at 64.0 GT/s with PAM-4 signaling: a low latency, high bandwidth, high reliability and cost-effective interconnect," in *Proceedings of the 2020 Annual Symposium on High-Performance Interconnects*, 2020, pp. 1–8.

[51] D. D. Sharma, "Compute Express Link®: An open industry-standard interconnect enabling heterogeneous data-centric computing," in *Proceedings of the 2022 Annual Symposium on High-Performance Interconnects*, 2022, pp. 5–12.

[52] J. Shun and G. E. Blelloch, "Ligra: a lightweight graph processing framework for shared memory," in *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2013, pp. 135–146.

[53] V. Srinivasan, D. M. Brooks, M. Gschwind, P. Bose, V. V. Zyuban, P. N. Strenski, and P. G. Emma, "Optimizing pipelines for power and performance," in *Proceedings of the 35th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2002, pp. 333–344.

[54] Standard Performance Evaluation Corporation, "SPEC CPU2017 Benchmark Suite." [Online]. Available: http://www.spec.org/cpu2017/

[55] P. Stanley-Marbell, V. C. Cabezas, and R. P. Luijten, "Pinned to the walls: impact of packaging and application properties on the memory and power walls," in *Proceedings of the 2011 International Symposium on Low Power Electronics and Design*, 2011, pp. 51–56.

[56] StorageReview, "4th Gen AMD EPYC Review (AMD Genoa)," 2022. [Online]. Available: https://www.storagereview.com/review/4th-gen-amd-epyc-review-amd-genoa

[57] Y. Sun, Y. Yuan, Z. Yu, R. Kuper, I. Jeong, R. Wang, and N. S. Kim, "Demystifying CXL memory with genuine cxl-ready systems and devices," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2023.

[58] TechPowerUp, "AMD "Matisse" and "Rome" IO Controller Dies Mapped Out," 2020. [Online]. Available: https://www.techpowerup.com/266287/amd-matisse-and-rome-io-controller-dies-mapped-out

[59] The Next Platform, "IBM POWER Chips Blur the Lines to Memory and Accelerators," 2018. [Online]. Available: https://www.nextplatform.com/2018/08/28/ibm-power-chips-blur-the-lines-to-memory-and-accelerators/#:~:text=The%20Centaur%20memory%20adds%20about

[60] The Register, "CXL absorbs OpenCAPI on the road to interconnect dominance," 2022. [Online]. Available: https://www.theregister.com/2022/08/02/cxl_absorbs_opencapi/

[61] D. Ustiugov, A. Daglis, J. Picorel, M. Sutherland, E. Bugnion, B. Falsafi, and D. N. Pnevmatikatos, "Design guidelines for high-performance SCM hierarchies," in *Proceedings of the 2018 International Symposium on Memory Systems (MEMSYS)*, 2018, pp. 3–16.

[62] S. Volos, "Memory Systems and Interconnects for Scale-Out Servers," Ph.D. dissertation, EPFL, Switzerland, 2015.

[63] S. Volos, D. Jevdjic, B. Falsafi, and B. Grot, "Fat Caches for Scale-Out Servers," *IEEE Micro*, vol. 37, no. 2, pp. 90–103, 2017.

[64] H. Wang, C.-J. Park, G. Byun, J. H. Ahn, and N. S. Kim, "Alloy: Parallel-serial memory channel architecture for single-chip heterogeneous processor systems," in *Proceedings of the 21st IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 2015, pp. 296–308.

[65] D. H. Yoon, J. Chang, N. Muralimanohar, and P. Ranganathan, "BOOM: Enabling mobile memory based low-power server DIMMs," in *Proceedings of the 39th International Symposium on Computer Architecture (ISCA)*, 2012, pp. 25–36.

[66] V. Young, S. Kariyappa, and M. K. Qureshi, "Enabling Transparent Memory-Compression for Commodity Memory Systems," in *Proceedings of the 25th IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 2019, pp. 570–581.

[67] Q. Zhu, S. Venkataraman, C. Ye, and A. Chandrasekhar, "Package design challenges and optimizations in density efficient (Intel® Xeon® processor D) SoC," in *2016 IEEE Electrical Design of Advanced Packaging and Systems (EDAPS)*, 2016.

# Appendix: Artifact Description/Artifact Evaluation

## Artifact Description (AD)

### I. Overview of Contributions and Artifacts

#### A. Paper's Main Contributions

The paper proposes CoaXiaL, a radical server architecture redesign that replaces DDR memory interfaces with high-bandwidth CXL. In the context of relating the artifact description, the relevant contribution is:

$C_1$ Demonstrate that the performance of manycore servers running bandwidth-heavy workloads can significantly improve by replacing the conventional DDR-based memory system with CoaXiaL, a carefully designed interface based memory system.

#### B. Computational Artifacts

$A_1$ https://zenodo.org/doi/10.5281/zenodo.13329058

| Artifact ID | Contributions Supported | Related Paper Elements |
|---|---|---|
| $A_1$ | $C_1$ | Figures 5-9 |

### II. Artifact Identification

#### A. Computational Artifact $A_1$

*Relation To Contributions*

The artifact consists of our Champsim-based simulation infrastructure, and a set of run-dispatching and post-processing scripts. All evaluation results (that are used to demonstrate performance gain from CoaXiaL) are generated with this artifact.

*Expected Results*

CoaXiaL will show significantly improved performance over baseline DDR system by reducing contention at memory controller from limited bandwidth.

*Expected Reproduction Time (in Minutes)*

Downloading traces might take a few hours (depends on network bandwidth). Compiling the simulators takes less than a minute per configuration, with 8 configurations. Each experiment runs for about 6 hours on average, so recreating all 280 experiments requires 1,680 core-hours (approximately 1-2 days on a 64-core server). Recreating the 70 representative experiments requires about 420 core-hours (approximately 7 hours on a single 64-core server). Note that some experiments can take up to 12 hours.

*Artifact Setup (incl. Inputs)*

*1) Hardware dependencies:* The artifact requires many-core server(s) to run all configurations and workloads. There are 280 simulations stemming from 8 configurations with 35 workloads. As all workloads can run in parallel, it would take about 1-2 days of runtime on a 64-core server. The 70 representative simulations require about 6-12 hours of runtime on a 64-core server (the rest are sensitivity studies). At least 4GB of memory per core is required.

*2) Software dependencies:* Compilation requires gcc/ g++, cmake, and make. Launch scripts use Bash. Trace download is streamlined using Megatools utility, although they can also be downloaded using wget. The plotting scripts use Python (specifically, matplotlib library).

*3) Data sets:* SPEC2017, LIGRA, and PARSEC workload dynamic execution traces that are publicly accessible online. Few additional traces taken for this work (STREAM, Masstree, Kmeans) will also be uploaded to a public repo.

*4) Installation:* Please clone the repository (see URL in Section I-B) and follow the step-by-step instructions available in the README file.

*Artifact Execution*

The workflow setup includes downloading the execution traces, cloning simulator repositories, compiling simulator binaries, and making changes to run-scripts (either using helper-scripts or manually) as required. Once set up, experiments are launched in parallel (depending on compute resources). Finally, the simulation results are parsed and graphs are plotted to recreate relevant figures.

*Artifact Analysis (incl. Outputs)*

The artifact provides scripts to parse the simulation results to derive the normalized IPC, weighted speedup, cache miss-rate, or cache capacity loss metrics, as required. The relevant commands are provided in the README. Note that coalescing the data from different benchmarks into a single file is not currently automated. The Python scripts plot the relevant graphs. This artifact enables recreation of figures 5, 6, 7, 8, and 9.

## Artifact Evaluation (AE)

### A. Computational Artifact $A_1$

*Artifact Setup (incl. Inputs)*

Refer to Artifact Setup from Description. Installation guide is in the anonymous repo - which needs to be downloaded and unzipped for now.

*Artifact Execution*

Step by step guides are in the repo's READMEs. Order of execution is:

**building**

- Download availble traces (as per guided README.md)
- install(compile) DRAMSim3 - simply calling 'make' is usually all it takes.
- compile ChampSim

For compiling ChampSim, volunteer will have to update PATHS to DRAMSim and input files as per their environment, pointed in the README. Lines to be modified are tagged with #SCAE

**Running**

Go to SCRIPTS directory.

- python3 runall.py

Before calling the script, volunteer must update file paths to the Champsim directory and trace files, in champsim_run.py and runall.py. Lines to be modified are tagged with #SCAE. champsim_run.py launches a single simulation. runall.py finds the traces and launches the Baseline and CoaXiaL run.

**Collecting and Plotting Results**

- collect_stats.py finds IPC, memory latency and bandwidth from the output files, and generates collected_stats.csv
- generate_pickles.py takes the csv file and generates .pkl files to be used by plotting script
- plot_all.py takes the .pkl files to plot the speedup and difference in latency and bandwidth between baseline and CoaXiaL across traces.

*Artifact Analysis (incl. Outputs)*

The execution flow should result in plots (.png, .pdf) of the speedup and difference in latency and bandwidth between baseline and CoaXiaL across traces(benchmarks). We've simplified the plot (and the running/collecting process) for the volunteers. The results may not be identical to the plots presented in the submission, as for many benchmarks we take the geometric mean of multiple runs from the same benchmark into a single data point in the paper, but skip the step in this automated process.