Making Locality-aware GEMM Compatible with Page-granularity Placement on Chiplet GPUs

Euijun Chung, *Student Member, IEEE,* Jae Hyung Ju, *Student Member, IEEE,* and Hyesoon Kim, *Fellow, IEEE*

***Abstract*—Multi-chiplet GPUs serve each memory access either from the chiplet's nearby HBM stack or from a remote stack across the silicon interposer, and locality-aware data placement is a common way to keep accesses local. We show that this placement is often not achievable on current GPUs, since data is placed at page granularity (4 KB hardware interleaving or 2 MB GPU pages), while the locality-optimal per-chiplet slices are finer than, or not multiples of, the page granularity. For the 67% of representative LLM GEMMs in this regime, adopting the placement on current systems leaves 75% of HBM accesses remote on a four-chiplet GPU. To address this, we propose *Chiplet-Contiguous Layout* (CCL), a global memory layout that contiguously stores the data consumed by each chiplet, so that the locality-aware placement becomes realizable with 4 KB placement granularity, without changes to the operating system or hardware. Across representative GEMMs from Qwen 3 30B and Llama 3.1 70B, CCL reduces remote HBM traffic by 9.5× over 4 KB interleaving and by 2.2× over coarse locality-aware placement, improving energy efficiency by 17% and 9%, respectively.**

***Index Terms*—Multi-chiplet GPU, GEMM, data placement.**

I. Introduction

MULTI-CHIPLET GPUs scale compute and memory beyond the reticle limit by integrating multiple GPU chiplets and memory stacks in a single package [1], [2]. Figure 1 shows a memory request in such GPUs being served either by the chiplet's nearby HBM stack (*local* access) or by a remote HBM stack through the silicon interposer (*remote* access), which increases latency and energy consumption [3], [4]. To reduce the remote HBM traffic, prior work proposes locality-aware data placement by co-locating compute with the data it accesses [3], [5], [6]. Hardware redesigns also aim to reduce the remote access cost and are complementary to locality-aware placement approaches [7].

However, the locality-optimal placement for GEMM (general matrix multiplication) depends on the matrix shape and layout, and GPU memory systems cannot support such arbitrary data placement. The placement either splits a matrix into large contiguous chunks per chiplet (*coarse-grained interleaving*) or slices every row or column along the contiguous dimension and distributes them across chiplets (*fine-grained interleaving*) [5], [6]. Across the representative GEMMs in LLMs, we find that 67% of them minimize remote HBM traffic under fine-grained interleaving, with per-chiplet slices from 384 B to 7,168 B for popular LLMs. These slices are finer than, or not multiples of, the *page granularity* in today's GPUs, which is either 4 KB with hardware interleaving or 2 MB with GPU pages. We call this the *misalignment problem*. When the desired slice sizes do not match the system's page granularity, 75% of HBM accesses become remote on a four-chiplet GPU even with locality-aware placement (Figure 3). The misalignment cannot be resolved either by access reordering or by virtual-to-physical mapping at page granularity, since the desired row or column slices fragment across page boundaries.

The authors are with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: euijun@gatech.edu; jhju@gatech.edu; hyesoon@cc.gatech.edu).

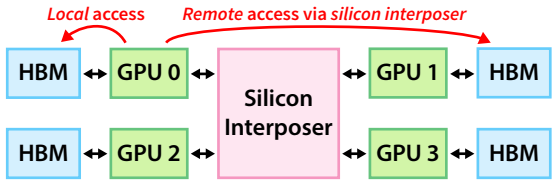

Fig. 1. Memory access on a 4-chiplet GPU. Remote access via a silicon interposer increases latency, power, and inter-chiplet bandwidth consumption.

To this end, we propose *Chiplet-Contiguous Layout* (CCL), a two-dimensional global memory layout that contiguously stores the data consumed by each chiplet, so the locality-aware placement for multiple GEMMs is achieved with a fixed (e.g., 4 KB) interleaving granularity. CCL requires no changes to the OS or hardware, and under the same 4 KB interleaving, it achieves lower remote HBM traffic compared to one-dimensional layouts (e.g., row-major, column-major) under locality-aware placements [6]. Across 72 representative GEMMs from Qwen 3 30B and Llama 3.1 70B, CCL reduces mean remote traffic by 7.2× on Qwen and 12.5× on Llama over 4 KB round-robin interleaving, and by 2.0× and 2.5× over coarse-grained locality-aware data placement. CCL also improves the energy efficiency of the GEMMs (TFLOP/J) by 17% and 9% on average over the two baselines, respectively.

II. Background and Motivation

A. Data Placement Granularity in Multi-chiplet GPUs

Locality-aware placement co-locates cooperative thread arrays (CTAs) with the data regions they access, using compiler analysis, profiling, or heuristics to infer CTA-to-data affinity [3], [5], [6], [8]. However, the memory system distributes data across chiplets only in fixed page-size units, e.g., 4 KB or 2 MB. Thus, current GPUs achieve optimal locality only when per-chiplet slices are multiples of the page size. The page-placement granularity is determined by hardware-level data interleaving (4 KB), which uses address bits to spread accesses across HBM stacks [1], or by 2 MB GPU pages.

B. GEMM Locality and the Misalignment Problem

GEMM is a key operation in LLM workloads and is a natural target for locality-aware optimizations on GPUs.

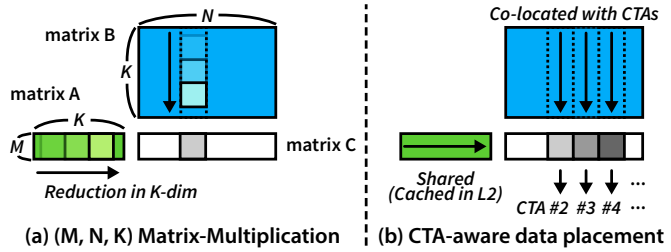

Fig. 2. (a) Tiled matrix-multiply. (b) Co-locating data with CTAs. When matrix $\mathbf{A}$ tiles are cached in each GPU chiplet's L2 cache, placing each tile column of matrix $\mathbf{B}$ near the CTAs allows HBM accesses to be local.

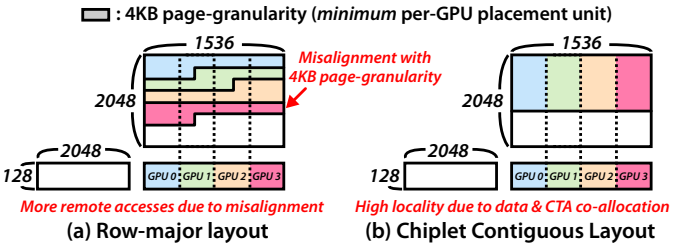

Fig. 3. Page-granularity placement of a GEMM on 4 chiplets. (a) Row-major layout: chiplet ownership does not align with 4 KB boundaries; a single page crosses chiplet boundaries and forces remote accesses. (b) Chiplet-Contiguous Layout: per-chiplet strips are contiguous and aligned with page boundaries.

We consider the canonical GEMM form $\mathbf{C} = \mathbf{AB}$. The BLAS extension to $\alpha\mathbf{AB} + \beta\mathbf{C}$ is straightforward, since the scalars do not affect data placement and CTA scheduling. Figure 2(a) depicts GEMM kernels decomposing $\mathbf{C}$ into output tiles, with each CTA computing one tile and streaming the corresponding $\mathbf{A}$ and $\mathbf{B}$ tiles along the K dimension. This tiled structure creates predictable CTA-to-data affinity where CTAs in the same row reuse the same $\mathbf{A}$ tiles. As shown in Figure 2(b), a locality-aware scheduler can exploit this structure by co-locating the corresponding tiles and CTAs on the same chiplet [6].

The locality-optimal placement for GEMM depends on the matrix dimensions and layout, and current multi-chiplet GPU memory systems cannot support such arbitrary data placement. Placement takes one of two forms: *coarse-grained interleaving* splits an operand into large contiguous chunks per chiplet, while *fine-grained interleaving* slices every row or column along the contiguous dimension and distributes the slices across chiplets. As shown in Figure 2, when $\mathbf{B}$ is in row-major layout, placing different column strips of $\mathbf{B}$ on different chiplets is fine-grained, as each row of $\mathbf{B}$ is split across chiplets. For a row width of N elements distributed across G chiplets, this requires an interleaving granularity of N/G elements, and N varies across matrices. Across the representative GEMMs, we find that 67% of them minimize remote HBM traffic under fine-grained interleaving.

For BF16 data and four chiplets, GEMMs in popular LLMs have row widths of 768, 4,096, and 14,336 elements, and these require row slices of 384 B, 2,048 B, and 7,168 B, respectively. However, when the desired interleaving granularity (row/column slices) is not a multiple of, or finer than, the page granularity of 4 KB or 2 MB, the *misalignment problem* leaves up to 75% of HBM accesses remote on a four-chiplet GPU even with locality-aware placement, as shown in Figure 3(a).

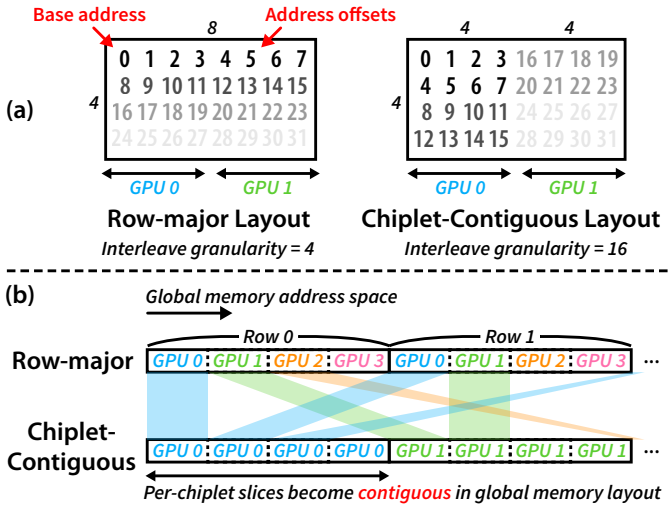

Fig. 4. Chiplet-Contiguous Layout (CCL) changes the global memory layout of a matrix. (a) Row-major layout makes rows contiguous, while CCL makes per-chiplet strips contiguous. (b) Row-major layout alternates chiplet slices within each row, whereas CCL scatters each row and groups per-chiplet slices.

Reordering accesses under the existing layout alone cannot solve this problem, as LLM GEMMs far exceed the L2 capacity, and the desired interleaving granularity exceeds cache line sizes (128 B). Virtual-to-physical remapping cannot change this either: pages move only as a whole, and a misaligned page contains slices consumed by multiple chiplets, so any mapping leaves the other chiplets' slices remote.

III. Chiplet-Contiguous Layout (CCL)

A. Remapping Row Interleaving into Contiguous Regions

Our key insight is to change the memory layout so that the data is expressed as contiguous chiplet-local regions, rather than fragmented row slices. We aim to pack the chiplet-local slices into contiguous regions whose sizes are multiples of, and whose boundaries are aligned to, the page granularity (e.g., 4 KB). As shown in Figure 3(b), by changing the memory layout so that the per-chiplet slices of $\mathbf{B}$ align with the page boundaries, the misalignment problem is eliminated and all CTA accesses to $\mathbf{B}$ become local.

Figure 4(a) illustrates Chiplet-Contiguous Layout (CCL) in comparison with row-major layout. Both represent the same logical matrix and the same per-chiplet column strip. In row-major layout, each row is stored contiguously, so the global memory order repeatedly alternates among slices for different chiplets within every row. For example, with two chiplets, each row's slices for both chiplets are interleaved in memory, requiring an interleaving granularity of 4 elements. In contrast, CCL first stores *all row slices consumed by chiplet 0*, then all row slices consumed by chiplet 1, and so on, raising the interleaving granularity to 16 elements. CCL comes in two variants, derived from row-major and column-major layouts; we illustrate only the row-major variant.

CCL supports both fine- and coarse-grained interleaving over the fixed 4 KB granularity. Unlike prior work [6], whose fine-grained placement must be rounded up to the fixed hardware granularity, CCL makes the desired fine-grained partition directly compatible with that granularity. For fine-grained

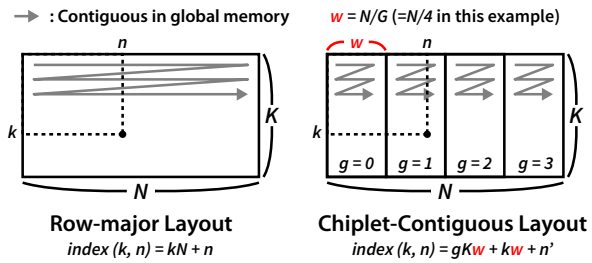

Fig. 5. Row-major and Chiplet-Contiguous address mapping for element (k, n) in a $K \times N$ matrix. Chiplet-Contiguous Layout decomposes n into a chiplet index g and local offset n'. w denotes the per-chiplet matrix width.

interleaving, each column strip forms a contiguous region aligned to 4 KB boundaries; for coarse-grained interleaving, consecutive rows form contiguous chunks within each strip.

B. Layout Mapping and Page-granularity Placement

We describe CCL's address mapping in Figure 5, using the $\mathbf{B}$ operand in Figure 2. For simplicity, we first describe the row-major variant of CCL. Let $\mathbf{B} \in \mathbb{R}^{K \times N}$ be distributed across G chiplets along the N dimension. We define the per-chiplet width as $w = N/G$ and decompose each column index $n \in \{0, \cdots, N-1\}$ into a chiplet index $g \in \{0, \cdots, G-1\}$ and an intra-chiplet column index $n' \in \{0, \cdots, N/G-1\}$:

$$g = \left\lfloor \frac{n}{w} \right\rfloor, \qquad n' = n \bmod w. \tag{1}$$

Here, g identifies the chiplet-local strip that contains column n, and n' identifies the column position within that strip.

In conventional row-major layout, element $\mathbf{B}[k, n]$ is stored at element index:

$$\text{index}_{\text{row}}(k, n) = k \cdot N + n. \tag{2}$$

This layout makes each row contiguous, but it also causes chiplet-local slices to repeat within every row. CCL instead stores data in chiplet-strip order:

$$\text{index}_{\text{CCL}}(k, n) = g \cdot K \cdot w + k \cdot w + n'. \tag{3}$$

All elements with the same chiplet index g therefore occupy one contiguous strip of $K \cdot w$ elements. The column-major variant is symmetric: with a per-chiplet height $v = K/G$, it decomposes the row index k into (g, k') and stores each chiplet's row strip of $N \cdot v$ elements contiguously. In both variants, strips align with 4 KB page boundaries when the strip size is a multiple of 4 KB. Either variant can be used for a given tensor. For simplicity, we match the variant to the tensor's original layout.

C. Software Support

Kernel software implements CCL's address mapping by modifying the global memory access logic. Frameworks such as CUTLASS/CuTe support such layout abstractions, enabling fast adoption of CCL in current kernel implementations. For example, for a matrix $\mathbf{B} \in \mathbb{R}^{K \times N}$ distributed across G chiplets, CCL is implemented by reshaping the logical view from (K, N) to $(K, G, N/G)$ with strides $(w, K \cdot w, 1)$.

We apply CCL to all matrix operands ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$) stored in global memory. Weights adopt CCL at model-load time and are reused across multiple GEMMs at runtime, so no extra pass over the data is added, and the on-disk checkpoint remains unchanged. Activations are produced directly by upstream kernels in CCL layout, which changes only the kernel's address computation. Translating between Eq. (2) and Eq. (3) adds only a few ALU operations per access, which are fully overlapped with HBM latency. Thus, there is no recurring rearrangement cost, and no repacking kernel runs between GEMMs.

IV. Evaluation

A. Methodology

We measure remote HBM traffic, performance, and energy consumption for representative LLM GEMMs under different data layouts and memory mappings.

Simulation framework. We use a custom tile-level GEMM locality simulator that models CTA execution, per-chiplet L2 caches, and HBMs, with a roofline-based performance model [9]. Each CTA computes one output tile and issues operand accesses according to the GEMM shape and CTA traversal order. The simulator classifies L2 misses as local or remote HBM accesses based on the data layout and memory-mapping policy. Since a GEMM accesses each operand region in a fixed, balanced pattern across chiplets, dynamically migrating pages only shifts remote accesses to another chiplet rather than eliminating them. Therefore, we do not model page migration.

Hardware configuration. We use an MI300X-class GPU configuration with four memory-locality domains (chiplets), each comprising two Accelerator Compute Dies (XCDs), for a total of 304 compute units (CUs). Each domain has an 8 MB private L2 cache with 128 B cache lines and 16-way associativity. GEMM kernels use 128×128 output tiles.

Workloads. We evaluate the GEMMs in QKVO (query, key, value, and output) projection and FFN (feed-forward network) of Qwen 3 30B (mixture-of-experts) and Llama 3.1 70B (dense) models on forward and backward passes. To run multiple GEMM configurations in LLM training, we sweep three batch sizes with token count 4K, 8K, 16K, yielding 72 GEMMs in total. We assume BF16 for all parameters. We group these GEMMs by their locality-optimal placement: *coarse-optimal* GEMMs, where partitioning every operand into large contiguous chunks minimizes remote HBM traffic, and *fine-optimal* GEMMs, where only fine-grained interleaving does so; 48 of the 72 (67%) GEMMs are *fine-optimal*. Extending CCL to attention kernels is our future work.

Baselines. We compare CCL against fixed-granularity round-robin interleaving at 4 KB, 64 KB, and 2 MB, which assigns consecutive fixed-size physical memory regions to chiplets in address order. We also include coarse locality-aware placement (Coarse LA) [6], which applies coarse-grained interleaving to every GEMM, splitting each matrix into G large contiguous chunks, one per chiplet. CCL instead selects coarse- or fine-grained interleaving per GEMM. For every scheme, we sweep CTA traversal and output-partition choices and report the configuration with the lowest remote HBM traffic.

Metrics. Remote HBM traffic reports remote operand reads that miss in L2, and remote output writes. We normalize

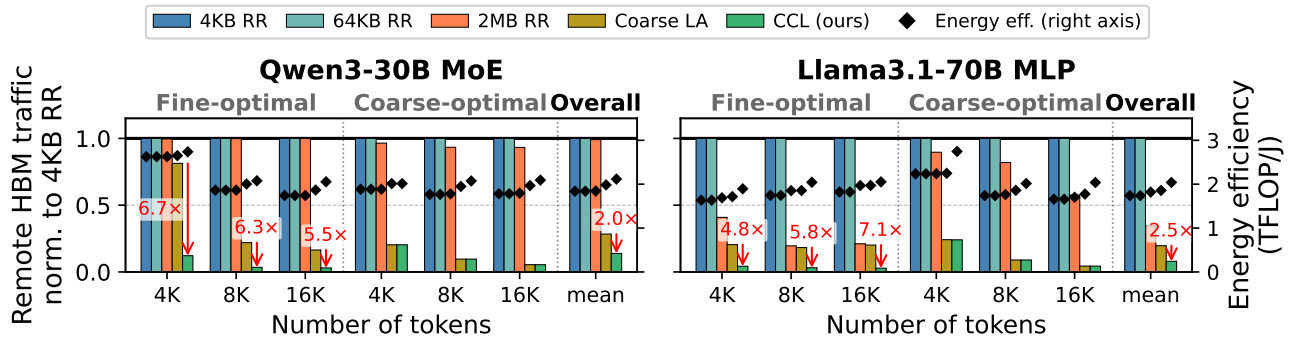

Fig. 6. Remote HBM traffic (bars) and energy efficiency (diamonds), for Qwen 3 30B (left) and Llama 3.1 70B (right). GEMMs are divided into two groups (fine-/coarse-optimal) depending on their optimal locality-aware placement strategy. Each group is swept over token counts of 4K, 8K, and 16K.

HBM traffic to the 4 KB round-robin baseline: each group in Figure 6 sums traffic over its GEMMs, and the mean group is the geometric mean of per-GEMM ratios. Energy efficiency is derived from per-access dynamic energy for an MI300X-class GPU: 0.3 pJ/FLOP for BF16 compute, 3.5 pJ/bit and 4.9 pJ/bit for local and remote HBM access, respectively [1]. The GEMMs are compute-bound at our token counts, so the benefit of CCL appears as energy savings per unit of work.

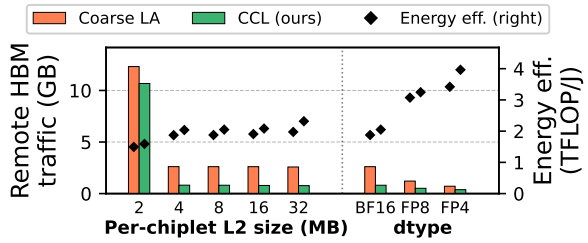

Fig. 7. Remote HBM traffic and energy efficiency on per-chiplet L2 capacity and data type sweep. Average across GEMMs with 16K tokens is reported.

B. Remote HBM Traffic and Energy Results

The results support our primary claim that CCL reduces remote HBM traffic compared to fixed-granularity and Coarse LA placement across LLM GEMMs. Figure 6 shows that changing fixed-granularity interleaving from 4 KB to 2 MB does not consistently reduce remote HBM traffic, because the placement unit still does not match GEMM-dependent chiplet-local slices. Coarse LA reduces traffic on some GEMMs when partitioning aligns with the GEMM access pattern, i.e., on *coarse-optimal* GEMMs, but on *fine-optimal* GEMMs, it leaves remote HBM traffic up to 16.5× higher than CCL on Qwen and 17.0× on Llama.

In contrast, CCL makes each chiplet's data contiguous, so locality-aware placement at 4 KB page granularity achieves fine-grained locality. CCL reduces mean remote HBM traffic to 13.9% of the 4 KB round-robin baseline for Qwen and 8.0% for Llama, corresponding to 7.2× and 12.5× reductions. Compared to Coarse LA, CCL further reduces it by 2.0× on Qwen and 2.5× on Llama. On energy efficiency, CCL achieves 15% (Qwen) and 17% (Llama) higher TFLOP/J than 4 KB round-robin (RR), and 7% and 10% higher than Coarse LA. CCL achieves higher energy efficiency on both fine- and coarse-optimal GEMMs, since its CTA-to-chiplet mapping increases L2 reuse, raising the mean L2 hit rate on Llama from 66% to 86%. These results show that CCL achieves better locality by eliminating the mismatch between the fixed page granularity and the optimal data placement.

We test whether CCL's locality advantage holds across per-chiplet L2 capacities and operand data types. Figure 7 shows that CCL remains below Coarse LA both when sweeping L2 capacity with BF16 fixed and when sweeping data type with the per-chiplet L2 fixed at 8 MB. CCL sustains 6–17% higher energy efficiency than Coarse LA across both sweeps.

V. Conclusion

Multi-chiplet GPUs require data locality, but achieving it for GEMMs often requires GEMM-dependent placement finer than, or misaligned with, a page, which conventional layouts cannot express. CCL groups data consumed by the same chiplet into contiguous global memory regions compatible with 4 KB interleaving, achieving fine-grained locality without memory-system changes. On LLM GEMMs, CCL reduces remote HBM traffic and achieves higher energy efficiency than 4 KB interleaving and coarse locality-aware placement.

References

[1] Advanced Micro Devices, Inc., "AMD CDNA™ 3 Architecture: The All-New AMD GPU Architecture for the Modern Era of HPC and AI," 2025.

[2] "NVIDIA Blackwell Architecture Technical Brief: Powering the New Era of Generative AI and Accelerated Computing," NVIDIA Corporation, Technical Brief, Mar. 2024.

[3] A. Arunkumar, E. Bolotin, B. Cho, U. Milic, E. Ebrahimi, O. Villa, A. Jaleel, C.-J. Wu, and D. Nellans, "Mcm-gpu: Multi-chip-module gpus for continued performance scalability," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 320–332.

[4] S. Zhang, M. Naderan-Tahan, M. Jahre, and L. Eeckhout, "Characterizing multi-chip gpu data sharing," *ACM Transactions on Architecture and Code Optimization*, vol. 20, no. 4, pp. 1–24, 2023.

[5] H. Kim, R. Hadidi, L. Nai, H. Kim, N. Jayasena, Y. Eckert, O. Kayiran, and G. Loh, "Coda: Enabling co-location of computation and data for multiple gpu systems," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 3, pp. 1–23, 2018.

[6] M. Khairy, V. Nikiforov, D. Nellans, and T. G. Rogers, "Locality-centric data and threadblock management for massive gpus," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 1022–1036.

[7] H. SeyyedAghaei, M. Naderan-Tahan, M. Jahre, and L. Eeckhout, "Memory-centric mcm-gpu architecture," *IEEE Computer Architecture Letters*, vol. 24, no. 1, pp. 101–104, 2025.

[8] J. Park, S. Jang, O. Kwon, Y. Lee, and S. Hong, "Leveraging chiplet-locality for efficient memory mapping in multi-chip module gpus," in *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture*, 2025, pp. 1040–1057.

[9] E. Chung and H. Kim, "A fast locality simulator for gemm design-space exploration on multi-chiplet gpus," *arXiv preprint arXiv:2606.11716*, 2026.