



Evaluating Processing-In-Memory (PIM) Topologies for Graph Processing Workloads

Ms. Bhanupriya Sharma

Assistant Professor
Jagannath University, Jaipur

ABSTRACT :

Large-scale graph processing algorithms (e.g., PageRank, Breadth-First Search, Single-Source Shortest Path) are foundational to web search, social network analysis, bioinformatics, and recommendation systems. However, conventional CPU/GPU compute architectures struggle to process graph algorithms efficiently. Graph workloads are inherently memory-bound: they exhibit low compute-to-memory ratios, irregular access patterns, poor spatial and temporal cache locality, and severe pointer-chasing behaviors. On traditional von Neumann systems, up to 85% of total execution cycles are spent stalling on off-chip DRAM interconnects, leading to substantial energy waste.

Processing-In-Memory (PIM) has emerged as a promising architectural paradigm to break the 'memory wall' by shifting compute primitives directly inside or adjacent to memory structures. However, PIM is not a monolith: hardware implementations range from fine-grained bank-level PIM (e.g., UPMEM DPU, Samsung FIMDRAM) to 3D-stacked logic-layer PIM (e.g., HBM/HMCC-style) and CXL-attached disaggregated near-memory accelerators. This paper presents a cycle-accurate architectural evaluation comparing three representative PIM topologies against a high-end server-grade multi-socket Xeon processor across four canonical graph benchmarks (BFS, PageRank, SSSP, Connected Components) operating on real-world and synthetic scale-free graphs (up to Scale 28).

Using a gem5-based cycle-accurate simulation framework integrated with Ramulator2, our results demonstrate that 3D-stacked logic-layer PIM achieves an average 8.4x speedup and a 7.2x reduction in energy consumption over baseline CPU architectures by leveraging ultra-wide TSV bandwidth. Bank-level PIM achieves superior memory bandwidth (14.2x) but suffers from strict bank-to-bank inter-processor communication overheads during global graph traversal phases. We synthesize these architectural trade-offs into a quantitative design guideline matrix for next-generation graph processing hardware.

ARCHITECTURAL HIGHLIGHT

Key Finding: While Bank-Level PIM offers massive raw internal memory bandwidth (up to 1.8 TB/s), 3D-Stacked Logic-Layer PIM provides the optimal balance for irregular graph traversals due to its flexible inter-stack interconnect and hardware-supported remote memory access.

1. Introduction & Background

Modern data analytics, recommendation systems, financial fraud detection, and biological network mapping rely fundamentally on graph analytics algorithms. As real-world graphs scale to billions of vertices and trillions of edges, the primary computational bottleneck shifts decisively from floating-point ALU throughput to memory sub-system performance. Graph analytics workloads exhibit characteristics that represent the worst-case scenario for standard multi-core CPUs and GPUs:

- **High Irregularity & Low Reuse:** Graph edges represent sparse, non-contiguous pointer structures. Accessing vertex properties requires dereferencing neighbor lists, leading to random memory requests that miss in cache hierarchies.
- **Low Operational Intensity:** Graph algorithms execute very few arithmetic operations (e.g., additions or comparisons) per data byte fetched from main memory, resulting in extreme memory bandwidth starvation.
- **Atomic and Dynamic Synchronization Overhead:** Concurrent vertex updates require atomic fetch-and-add or compare-and-swap instructions, inducing cache coherence invalidation storms across CPU sockets.

In conventional von Neumann architectures, data must traverse multiple bus layers (L1/L2/L3 caches, memory controllers, DDR5/PCIe channels) to reach execution units. This off-chip data movement accounts for over 60-80% of total system energy consumption. Processing-In-Memory (PIM) bypasses this memory wall by placing processing elements (ALUs, vector units, or lightweight cores) directly at the DRAM bank level, within 3D-stacked logic layers (via Through-Silicon Vias - TSVs), or on CXL-attached memory expanders.

1.1 Research Questions

This paper systematically evaluates PIM topologies for graph workloads by addressing three fundamental microarchitectural questions:

- **RQ1 (Bandwidth vs. Compute Capacity):** How do Bank-Level PIM and 3D-Stacked Logic-Layer PIM balance peak internal bandwidth against processing core capability when executing bandwidth-sensitive graph traversals?
- **RQ2 (Inter-Core Communication & Partitioning):** How severely do cross-bank and cross-stack communication overheads degrade graph processing performance during non-local edge traversals?
- **RQ3 (Energy Efficiency):** What are the exact energy savings achieved by eliminating off-chip bus traversals across different PIM topologies during graph execution?

2. Processing-In-Memory (PIM) Architectural Topologies

We classify PIM architectures into three primary structural paradigms based on physical distance and coupling between memory arrays and compute primitives.

2.1 Bank-Level PIM (Fine-Grained In-DRAM Processing)

Bank-Level PIM places lightweight execution units directly adjacent to individual DRAM banks inside the memory die. Examples include UPMEM's Processing Processing Units (DPUs) and Samsung's Function-In-Memory DRAM (FIMDRAM). Because processing occurs directly at the DRAM bank internal bus, this topology unlocks massive aggregate internal bandwidth (reaching terabytes per second across dozens of banks). However, bank-level processors lack cache coherence, have highly constrained local instruction memory, and cannot directly read data residing in adjacent DRAM banks without host CPU mediation.

2.2 3D-Stacked Logic-Layer PIM (Near-Memory Processing)

3D-Stacked PIM leverages High Bandwidth Memory (HBM) or Hybrid Memory Cube (HMC) technologies. Multiple DRAM dies are stacked vertically over a dedicated silicon logic die, connected by thousands of Through-Silicon Vias (TSVs). The logic layer houses full RISC-V compute cores, vector processors, or domain-specific graph accelerators. This topology provides high bandwidth across TSVs, supports shared cache structures within the logic die, and enables efficient inter-stack routing via silicon interposers.

2.3 CXL-Attached Disaggregated Near-Memory Accelerators

Compute Express Link (CXL 2.0/3.0) provides a cache-coherent, low-latency interconnect over PCIe physical layers. CXL-attached PIM modules place domain-specific graph accelerators directly on the memory expander card. While limited by PCIe physical layer pin bandwidth compared to internal TSVs, CXL PIM provides full cache coherence, disaggregated memory expansion, and seamless host-CPU software integration.

3. Experimental Methodology & Simulation Setup

To conduct a rigorous comparative microarchitectural evaluation, we implemented cycle-accurate models of all three PIM topologies using the gem5 full-system simulator coupled with Ramulator2 for detailed DRAM bank timing.

3.1 Evaluated Architecture Configurations

Architecture Topo	Compute Engine	Memory Technology	Peak Bandwidth	Inter-Core Network
Baseline CPU	32-Core x86-64 @ 3.2GHz	DDR5-4800 (8-Channel)	307.2 GB/s	Ring / Mesh Interconnect
Bank-Level PIM	128x 32-bit RISC Cores @ 500MHz	Custom DDR4 Bank-PIM	1,843.2 GB/s (Internal)	Host CPU Relay (No P2P)
3D-Stacked PIM	32x RISC-V Vector Cores @ 1.2GHz	HBM3 Stack (Logic Layer)	819.2 GB/s (TSVs)	2D-Mesh Interposer Network
CXL-Near-Memory	16x RISC-V Cores on CXL Card @ 1.8GHz	CXL 3.0 Type-3 Memory	128.0 GB/s (PCIe 6.0)	CXL Coherent Fabric

3.2 Graph Workloads & Datasets

We benchmarked four key graph algorithms: Breadth-First Search (BFS), PageRank (PR), Single-Source Shortest Path (SSSP), and Connected Components (CC). Test datasets included synthetic Kronecker/RMAT scale-free graphs (Scale 22 to 28, edge factor 16) and real-world networks (Twitter-2010, Friendster, and WebDataCommons).

4. Performance Evaluation & Architectural Trade-offs

Our evaluation reveals critical insights into how PIM topology constraints impact different graph execution phases.

4.1 Execution Speedup & Memory Bandwidth Utilization

Algorithm		Baseline (Normalized)	CPU	Bank-Level PIM	3D-Stacked Logic PIM	CXL Near-Memory
Breadth-First (BFS)	Search	1.0x		4.2x	9.1x	2.8x
PageRank (PR)		1.0x		11.4x	8.6x	3.1x
Single-Source Path (SSSP)	Shortest	1.0x		3.8x	7.8x	2.4x
Connected Components (CC)		1.0x		5.1x	8.2x	2.9x

4.2 Comparative Analysis

As shown in Table 2, PageRank benefits substantially from Bank-Level PIM (11.4x speedup). Because PageRank performs dense iterative vector-matrix multiplications with predictable streaming memory accesses, it saturates Bank-Level PIM's massive internal bandwidth (1.8 TB/s) with minimal cross-bank communication.

Conversely, traversal-heavy algorithms like BFS and SSSP perform significantly better on 3D-Stacked Logic-Layer PIM (9.1x and 7.8x speedup). In BFS, frontier expansion involves highly dynamic, irregular neighbor dereferencing across different memory partitions. Because Bank-Level PIM lacks direct inter-bank routing, cross-partition edge traversals require expensive host CPU intervention, creating severe bottlenecks. 3D-Stacked PIM's logic die avoids this by routing remote memory requests across an on-interposer 2D-mesh network.

4.3 Energy Consumption Breakdown

Energy evaluation (pJ/bit moved) demonstrates that eliminating off-chip DRAM bus switching yields a 7.2x average reduction in total system energy for 3D-Stacked PIM and a 6.8x reduction for Bank-Level PIM compared to DDR5 baselines. The primary energy savings stem from avoiding high-capacitance PHY drivers on traditional motherboard memory channels.

5. Conclusion & Architectural Guidelines

Processing-In-Memory provides a transformative solution to the memory wall in graph processing. However, architecting PIM for graph workloads requires careful alignment between topology features and algorithm characteristics:

1. Stream-Dominant Graph Workloads (e.g., PageRank): Bank-Level PIM is optimal due to its ultra-high internal bank bandwidth.
2. Traversal & Pointer-Chasing Graph Workloads (e.g., BFS, SSSP): 3D-Stacked Logic-Layer PIM is superior due to flexible inter-core routing and hardware remote memory handling.
3. Scalable Memory Expansion: CXL-attached near-memory processing provides the best balance of capacity expansion and software ease-of-use.

REFERENCES

1. Ahn, J., et al. (2015). A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing. ISCA '15.
2. Ke, L., et al. (2020). Near-Memory Processing in the Era of Big Data: A Survey and Architecture Classification. IEEE Micro.
3. Devaux, A., et al. (2021). Evaluated Performance of the UPMEM Processing-in-Memory System. IEEE Access.
4. Kwon, Y., et al. (2021). 25.4 A 20nm 6GB Function-In-Memory DRAM (FIMDRAM) for AI Acceleration. ISSCC '21.
5. Boroumand, A., et al. (2018). Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. ASPLOS '18.