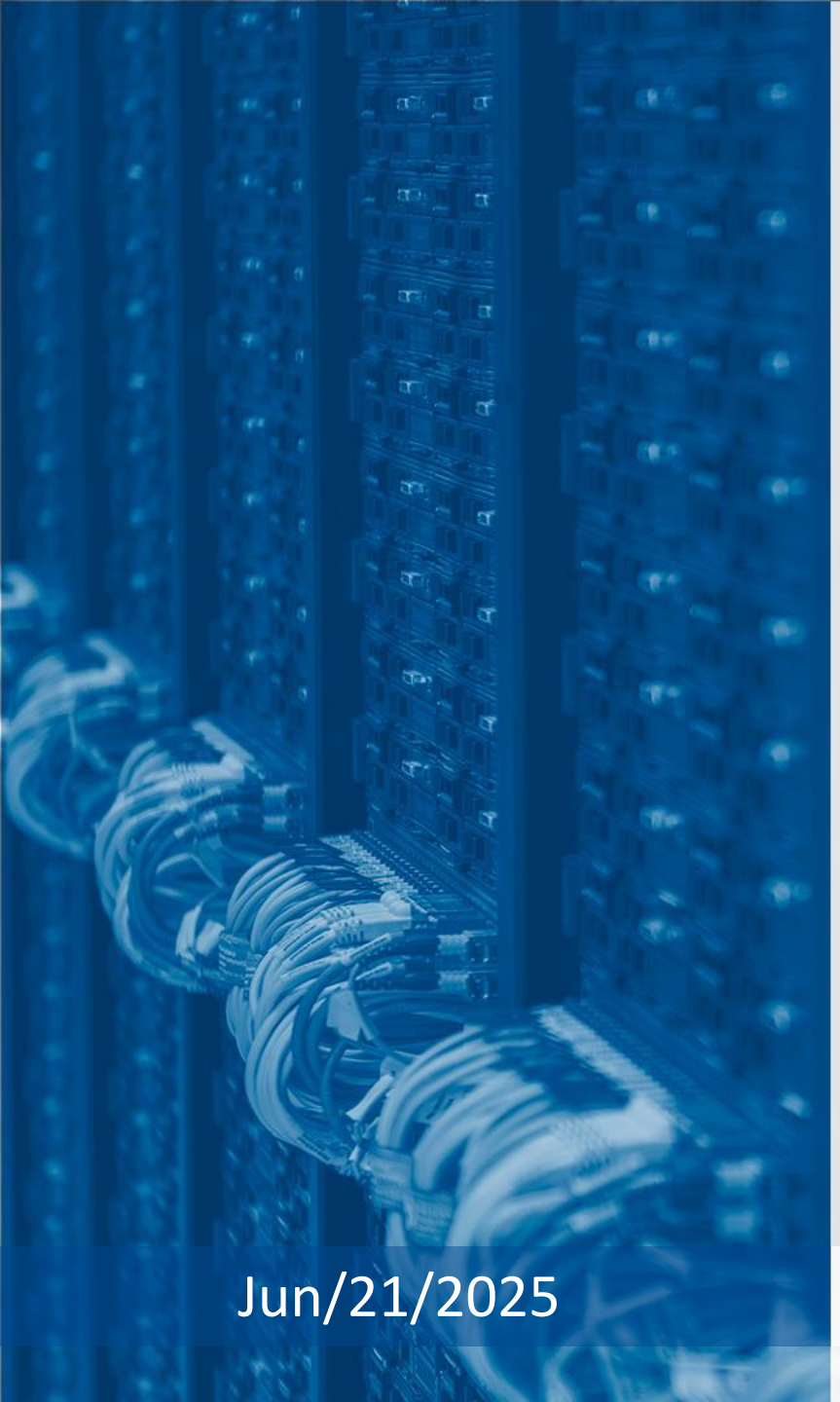


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Hardware-Software Co-designed Near-Cache Accelerator for Graph Pattern Mining

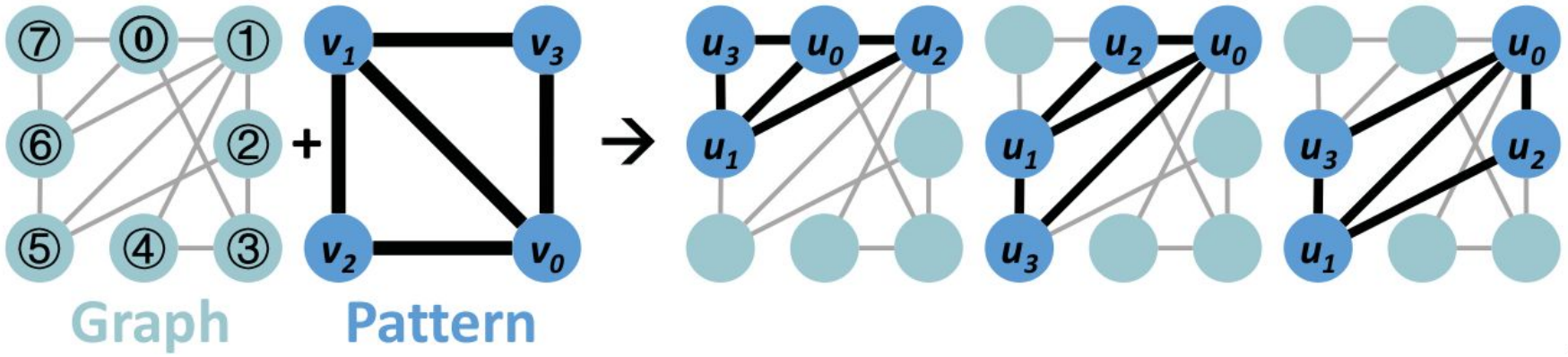
Julian Pavon, Ivan Vargas Valdivieso Osman Unsal and Adrian Cristal

Jun/21/2025

Barcelona Supercomputing Center

Graph pattern mining

Graph Pattern Mining (GPM) is the process of discovering frequent, interesting, or relevant subgraph structures within a larger graph dataset.



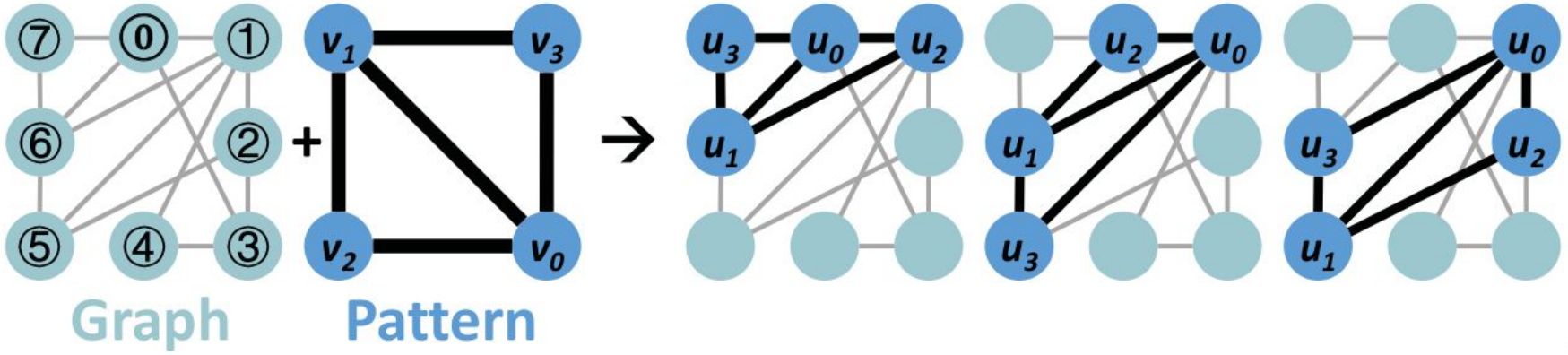
Dai, G., Zhu, Z., Fu, T., Wei, C., Wang, B., Li, X., ... & Wang, Y. (2022, June). Dimmining: pruning-efficient and parallel graph mining on near-memory-computing. In Proceedings of the 49th Annual International Symposium on Computer Architecture (pp. 130-145).

Graph pattern mining

Graph Pattern Mining (GPM) is the process of discovering frequent, interesting, or relevant subgraph structures within a larger graph dataset.

It helps identify:

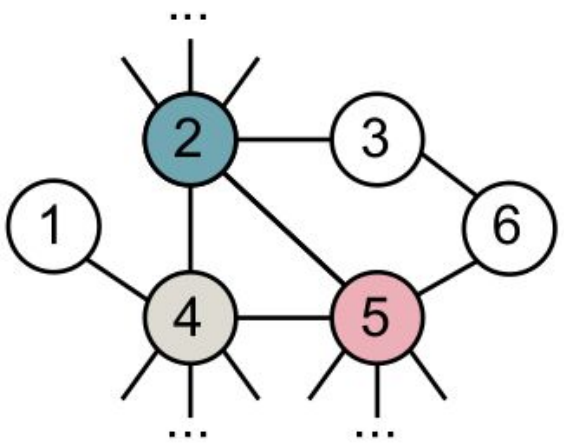
1. Common interaction patterns (e.g., in social networks)
2. Functional motifs (e.g., in biological networks)
3. Structural anomalies or rules (e.g., in knowledge graphs or fraud detection)



Dai, G., Zhu, Z., Fu, T., Wei, C., Wang, B., Li, X., ... & Wang, Y. (2022, June). Dimmining: pruning-efficient and parallel graph mining on near-memory-computing. In Proceedings of the 49th Annual International Symposium on Computer Architecture (pp. 130-145).

GPM's bottlenecks

The most time consuming operation in GPM algorithms is index matching.

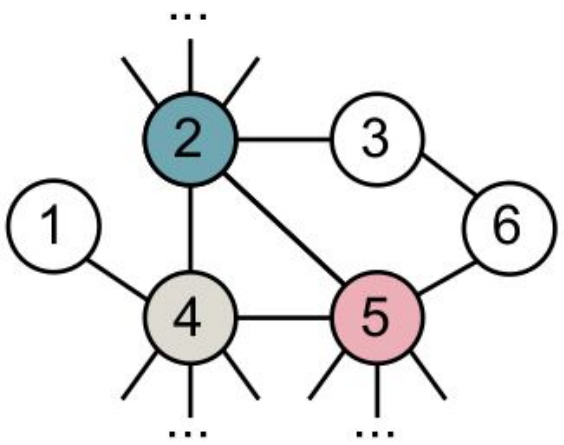


N	Neig. List
1	[4]
2	[3, 4, 5]
3	[2, 6]
4	[1, 2, 5]
5	[2, 4, 6]
6	[3, 5]

GPM's bottlenecks

The most time consuming operation in GPM algorithms is index matching.

1. Index matching incurs in cache pollution, as a large portion of the loaded elements are discarded, leading to inefficient memory utilization.
2. They incur in hard-to-predict divergence control (e.g., frequent if-else branches), which reduces the effectiveness of branch predictors.



N	Neig. List
1	[4]
2	[3, 4, 5]
3	[2, 6]
4	[1, 2, 5]
5	[2, 4, 6]
6	[3, 5]

Near-data processing

Prior work has explored various approaches to improve the performance of commercial CPUs for GPM algorithms, one of the most prominent is **Near-Data Processing (NDP)** architectures.

Near-data processing

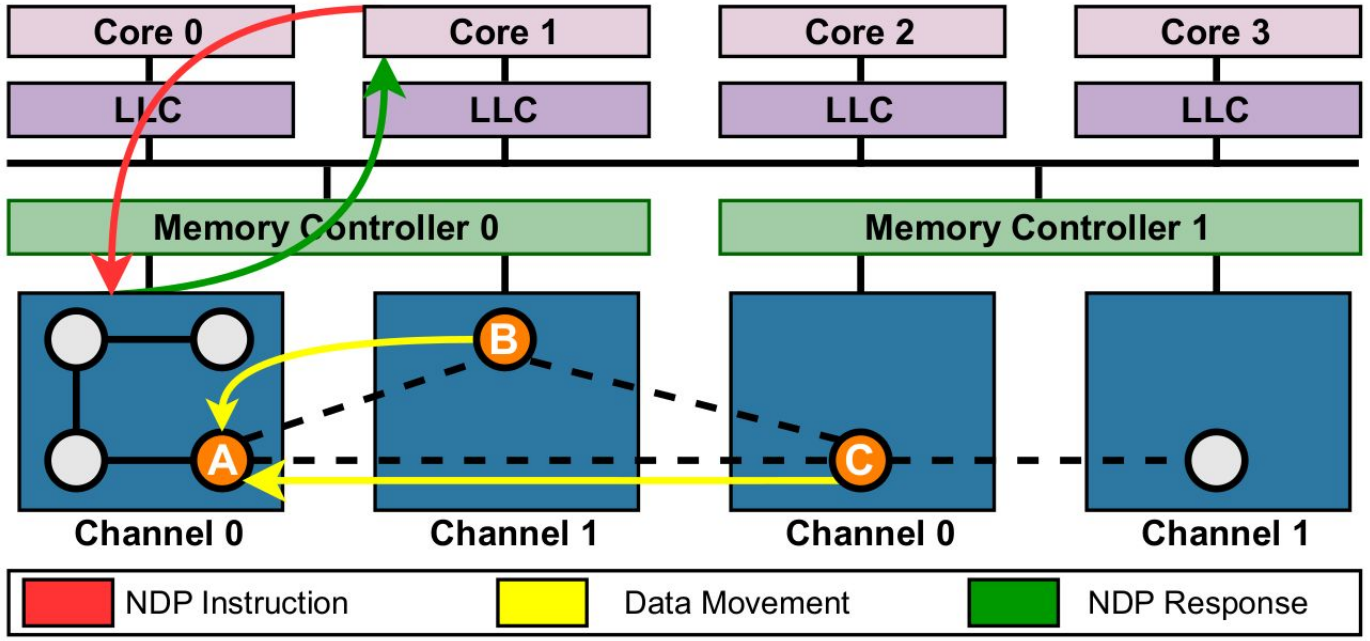
Prior work has explored various approaches to improve the performance of commercial CPUs for GPM algorithms, one of the most prominent is **Near-Data Processing (NDP)** architectures.

NDP designs execute index matching directly in main memory (e.g., DRAM), reducing data movement and cache pollution.

Near-data processing

However, NDP has key limitations:

1. it struggles to exploit the locality in GPM algorithms because of the interleaved data distribution across memory channels in commercial CPUs.

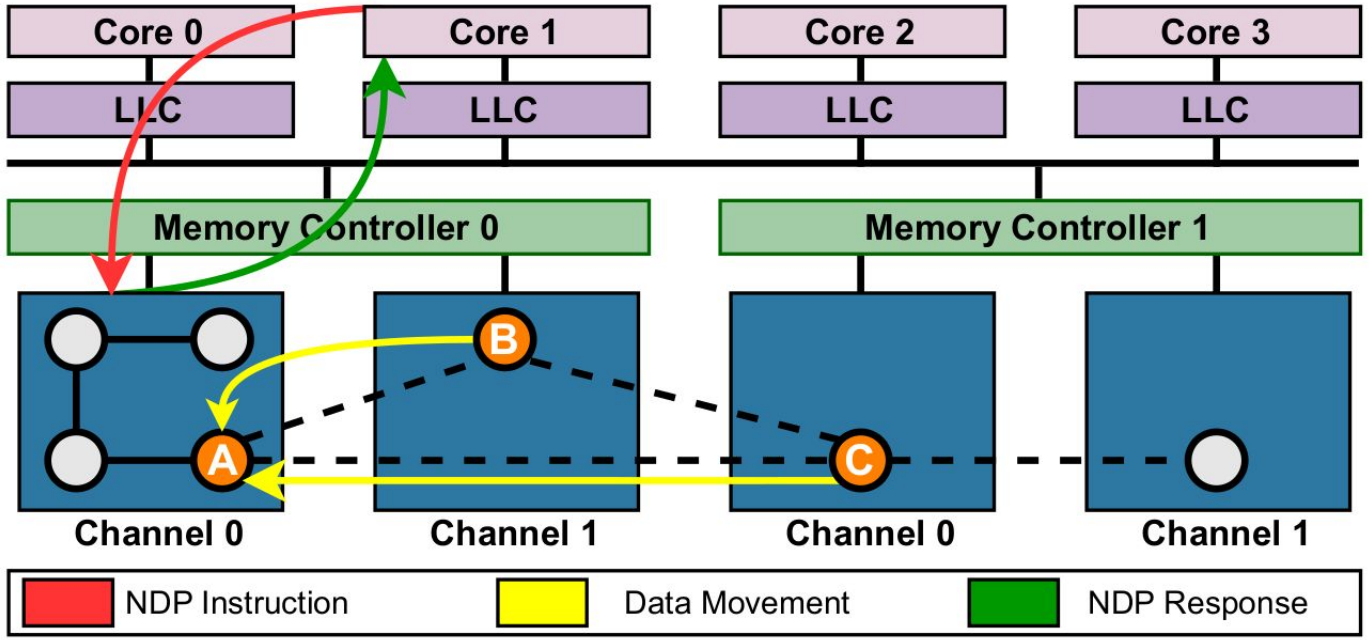


Current approaches

However, NDP has key limitations:

1. it struggles to exploit the locality in GPM algorithms because of the interleaved data distribution across memory channels in commercial CPUs.

2. It relies on physically contiguous data, which has limited scalability due to fragmentation and the scarcity of large contiguous memory blocks in real HPC systems and data center systems.



Desired features

A general purpose system to accelerate GPM algorithms should include the following features:

1. Reduce cache pollution and hard-to-predict divergence control.
2. Support virtual memory mapping.

Near-cache processing

This work explores Near-Cache Processing (NCP) as a means to improve GPM performance on commercial multi-core CPUs, motivated by two key insights.

Near-cache processing

This work explores Near-Cache Processing (NCP) as a means to improve GPM performance on commercial multi-core CPUs, motivated by two key insights.

1. GPM algorithms inherently exhibit data locality, but cache pollution diminishes its benefits. Processing data directly in lower cache levels, such as the Last-Level Cache (LLC), helps mitigate cache pollution in upper levels.

Near-cache processing

This work explores Near-Cache Processing (NCP) as a means to improve GPM performance on commercial multi-core CPUs, motivated by two key insights.

1. GPM algorithms inherently exhibit data locality, but cache pollution diminishes its benefits. Processing data directly in lower cache levels, such as the Last-Level Cache (LLC), helps mitigate cache pollution in upper levels.

2. NCP can be integrated with existing virtual memory support, removing the need of physically contiguous memory.

Near-cache processing

Moving processing to the LLC

Index matching operations generate transient data into the cache, evicting useful vertex and neighbor lists before they can be fully exploited, which reduces locality and increases memory access overhead.

Near-cache processing

Moving processing to the LLC

Index matching operations generate transient data into the cache, evicting useful vertex and neighbor lists before they can be fully exploited, which reduces locality and increases memory access overhead.

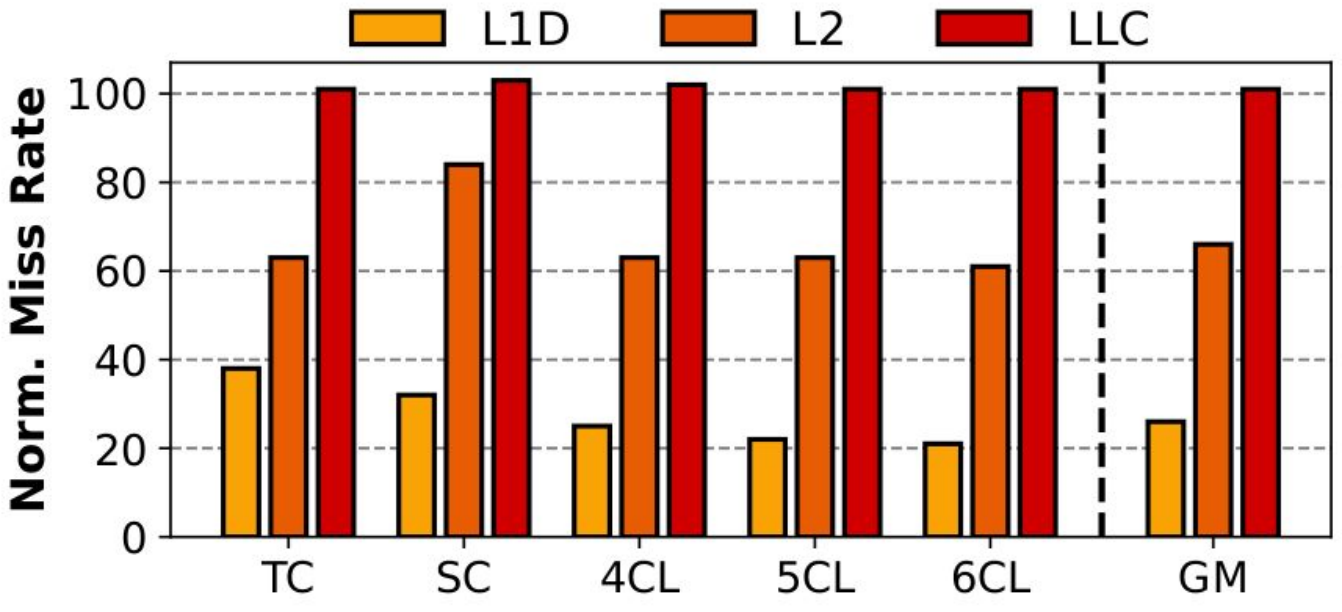
Computing index matching in the LLC prevents index matching transient data from evicting critical vertex data in upper cache levels.

Near-cache processing

Moving processing to the LLC

Index matching operations generate transient data into the cache, evicting useful vertex and neighbor lists before they can be fully exploited, which reduces locality and increases memory access overhead.

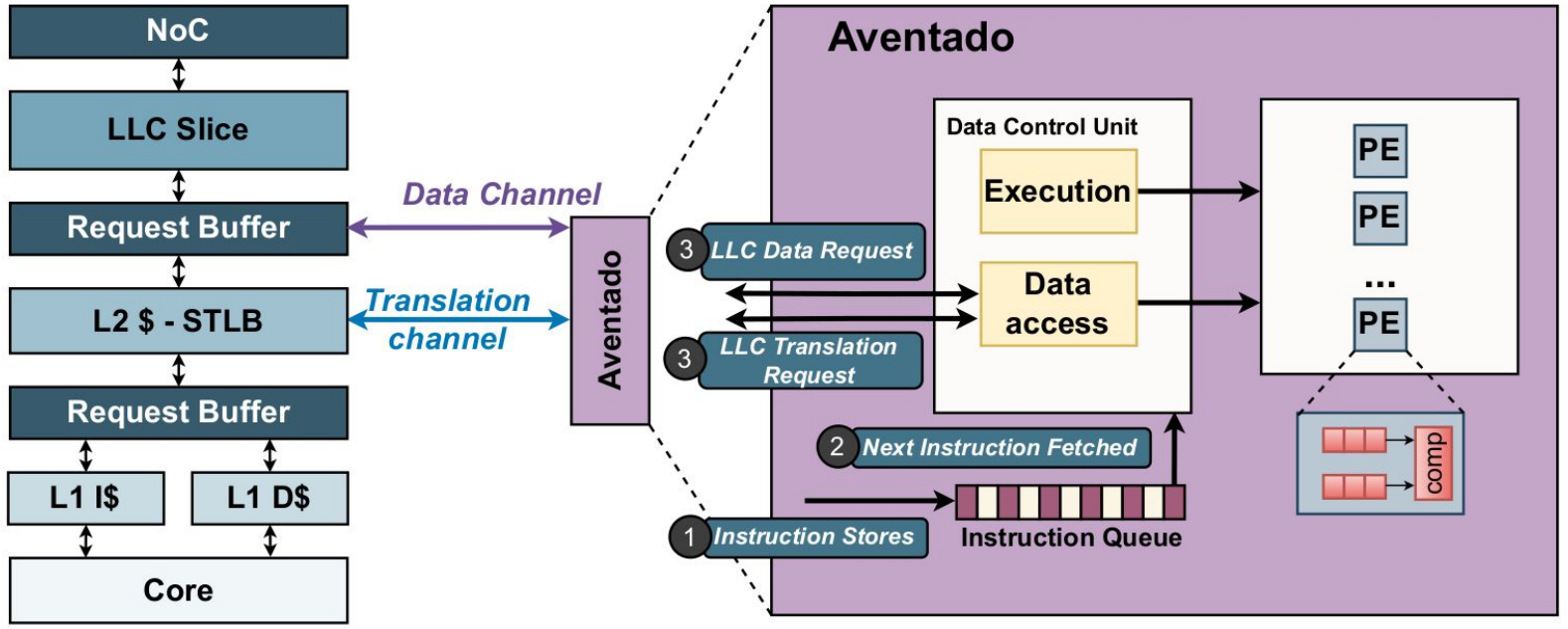
Computing index matching in the LLC prevents index matching transient data from evicting critical vertex data in upper cache levels.



Aventado overview

We introduce Aventado, a hardware-software co-designed near-LLC accelerator carefully designed to accelerate index matching operations.

1. Aventado works as a decoupled programmable unit.
2. Includes virtual memory support by reusing the cache hierarchy TLBs, avoiding the need for dedicated memory mappings.



Aventado ISA

Aventado's ISA is composed of two instruction to execute Intersection and difference between two sets of neighbors.

Instruction	Functionality
aventado_intersect *[addr0], len0, [addr1], len1*	Intersects two input sets
aventado_difference *[addr0], len0, [addr1], len1*	Differences two input sets

Aventado ISA

Example

Triangle counting

```
1: define triangleCounting(Graph G)
2: num_triangles = 0
3: preprocessing_prune_graph(G)
4: for each u0 in G.nodes do
5:     Nu0 = G.pruned_neighbors_list[u0]
6:     for each u1 in Nu0 do
7:         Nu1 = G.pruned_neighbors_list[u1]
8:         Nu0,u1 = intersection(Nu0, Nu1)
9:         num_triangles += Nu0,u1.size()
10: return num_triangles
```

Triangle counting using Aventado

```
1: define triangleCounting(Graph G)
2: num_triangles = 0
3: preprocessing_prune_graph(G)
4: for each u0 in G.nodes do
5:     Nu0 = G.pruned_neighbors_list[u0]
6:     for each u1 in Nu0 do
7:         Nu1 = G.pruned_neighbors_list[u1]
8:         Nu0,u1 = aventado_intersect(Nu0, Nu0.size(),
                                       Nu1, Nu1.size())
9:         num_triangles += Nu0,u1.size()
10: return num_triangles
```

Evaluation methodology

Simulator:

We model and evaluate Aventado using an in-house, cycle-accurate simulator that simultaneously runs both functional and microarchitectural performance simulation on a workload.

Evaluation methodology

Simulator:

We model and evaluate Aventado using an in-house, cycle-accurate simulator that simultaneously runs both functional and microarchitectural performance simulation on a workload.

CPU:

The simulated system consists of 16 Neoverse-N1-like out-of-order cores, three levels of cache (1MB LLC-slice per core) and 4 HBM2e memory channels. Each core features an Aventado module.

Evaluation methodology

Simulator:

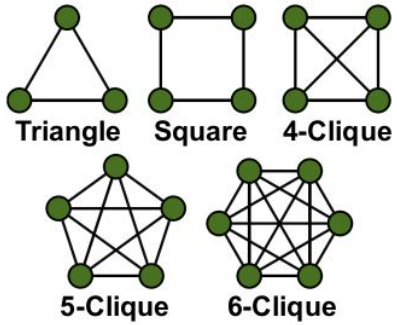
We model and evaluate Aventado using an in-house, cycle-accurate simulator that simultaneously runs both functional and microarchitectural performance simulation on a workload.

CPU:

The simulated system consists of 16 Neoverse-N1-like out-of-order cores, three levels of cache (1MB LLC-slice per core) and 4 HBM2e memory channels. Each core features an Aventado module.

Benchmarks: (GAPBS+GraphPhi)

Input Patterns



Real-World Datasets Used for Evaluation

Graph	#Vtx	#Edge	Average Degree	Size
Wiki-Vote	7.1k	103.7k	14.6	0.5MB
Ca-Astro	18k	396k	21	2.2MB
Com-Youtube	1.1M	5.9M	5	38.7MB
Soc-Pokec	1.6M	30.6M	18	123.9 MB

Evaluation methodology

Comparison:

Together with Aventado, we evaluated NDMiner [1] and DIMMining [2], two NDP accelerators directly integrated into DIMM modules.

[1] Dai, G., Zhu, Z., Fu, T., Wei, C., Wang, B., Li, X., ... & Wang, Y. (2022, June). Dimmining: pruning-efficient and parallel graph mining on near-memory-computing. In Proceedings of the 49th Annual International Symposium on Computer Architecture (pp. 130-145).
[2] Talati, N., Ye, H., Yang, Y., Belayneh, L., Chen, K. Y., Blaauw, D., ... & Dreslinski, R. (2022, June). Ndminer: accelerating graph pattern mining using near data processing. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (pp. 146-159).

Evaluation methodology

Comparison:

Together with Aventado, we evaluated NDMiner [1] and DIMMining [2], two NDP accelerators directly integrated into DIMM modules.

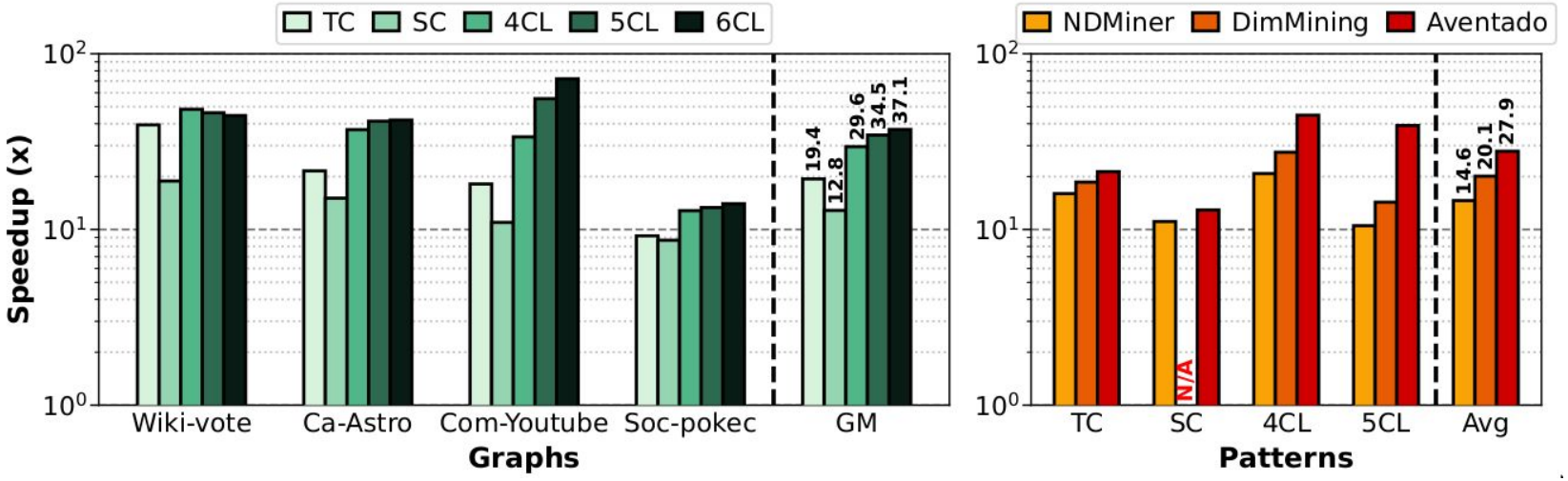
Area:

To evaluate the area impact of our design, we physically implemented all Aventado hardware modules using SystemVerilog and Synopsys' ICC2 Place and Route tool with a 7nm technology node. In our evaluation, we set the instruction queue to 384 bytes (16 instructions) and 8 PEs. Aventado adds 0.08mm2 extra per-core, resulting in a negligible area overhead of 0.9% and 0.3% compared to the baseline core and SoC (including a Aventado instance per core), respectively.

[1] Dai, G., Zhu, Z., Fu, T., Wei, C., Wang, B., Li, X., ... & Wang, Y. (2022, June). Dimmining: pruning-efficient and parallel graph mining on near-memory-computing. In Proceedings of the 49th Annual International Symposium on Computer Architecture (pp. 130-145).
[2] Talati, N., Ye, H., Yang, Y., Belayneh, L., Chen, K. Y., Blaauw, D., ... & Dreslinski, R. (2022, June). Ndminer: accelerating graph pattern mining using near data processing. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (pp. 146-159).

Results

Performance results:



Area:

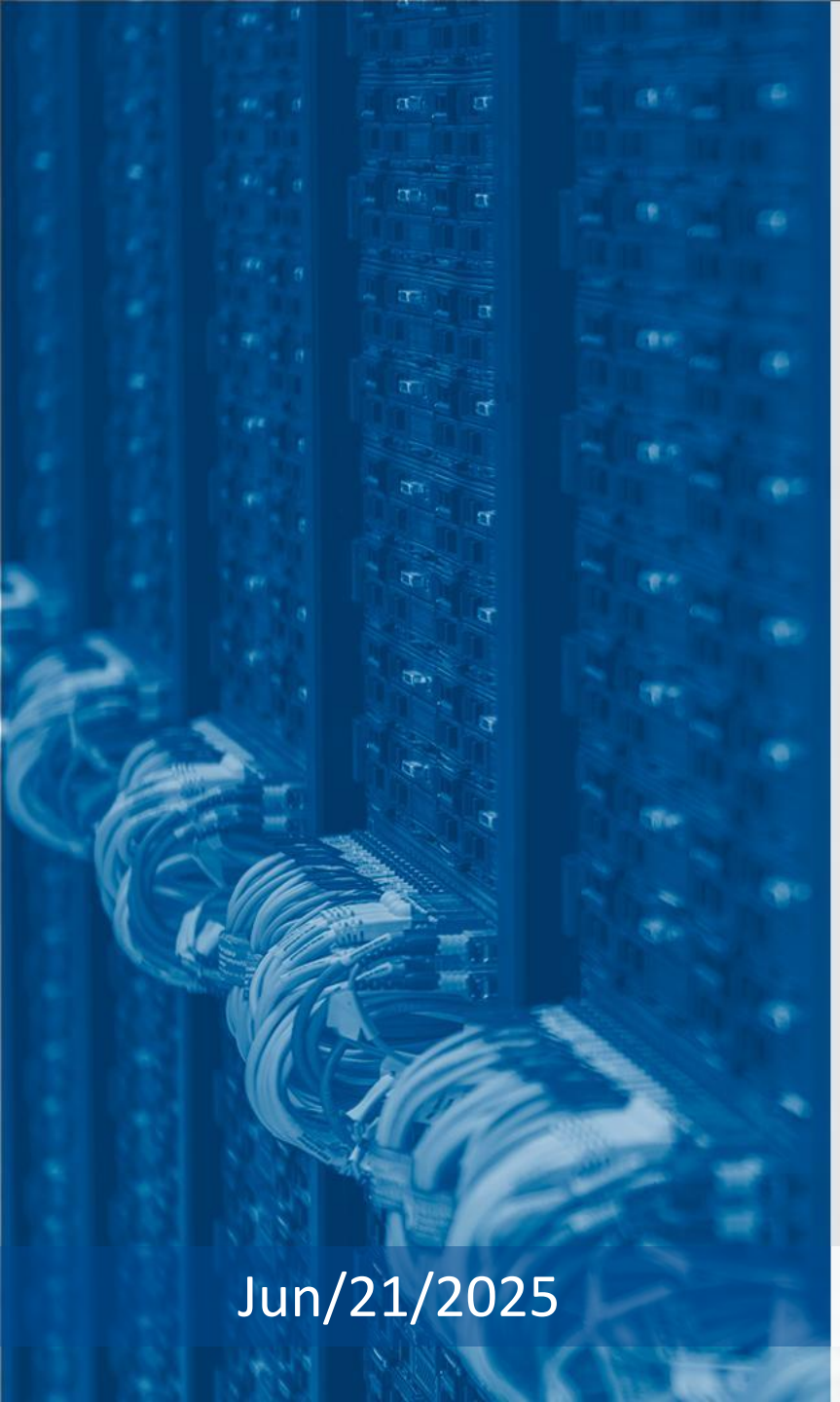
Aventado adds 0.08mm^2 extra per-core, resulting in a negligible area overhead of 0.9% and 0.3% compared to the baseline core and SoC (including an Aventado instance per core), respectively.

Conclusions

We present Aventado, a novel hardware-software co-designed near-cache accelerator that provides efficient hardware support for index matching operations in GPM workloads.

While Aventado is agnostic to any user-defined pattern, it is also adaptable to any general-purpose architecture.

Our evaluation shows that Aventado is highly area efficient (0.5% overhead) while providing 26.7× and 1.7× better performance than software and hardware baselines, respectively.



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Hardware-Software Co-designed Near-Cache Accelerator for Graph Pattern Mining

Julian Pavon, Ivan Vargas Valdivieso Osman Unsal and Adrian Cristal

Jun/21/2025

Barcelona Supercomputing Center