TGN-PNM: A Near-Memory Architecture for Temporal GNN Inference on 3D-Stacked Memory

Alif Ahmed, Felix Lin, Jundong Li, Kevin Skadron
University of Virginia

Presented by: Kevin Skadron

This work was supported in part by PRISM, one of seven centers in JUMP2.0, a Semiconductor Research Corporation program sponsored by MARCO and DARPA

Motivation

- Graph Neural Networks (GNNs) are widely used in social networks, bioinformatics, recommendation systems, etc.
- Many accelerators have been proposed for **static** GNNs:
  - HyGCN, AWB-GCN, GCoD, StreamGCN, FlowGCN, etc.
  - Offers orders-of-magnitude better inference throughput and latency over CPU/GPU.
- However, most real-world graphs are **dynamic** in nature, not static:
  - E.g., on social networks, users are forming new connections, deleting old ones, and groups are constantly evolving.
  - Prediction accuracy can be improved drastically by leveraging this ***temporal*** information → **temporal GNNs (TGNNs)**.
- Very challenging to design an accelerator for TGNNs. We found only prior accelerator for TGNNs → FPGA-based *tFGPA* [IPDPS'22].

Challenges in Designing a TGNN Accelerator (1)

- No standard TGNN model architecture.
  - Graph Convolutional Network (GCN) is the standard model architecture for static GNNs. However, there is no such standard model architecture for TGNNs.
  - Preferred architecture boils down to accuracy-complexity tradeoff.
- Cannot be easily decomposed into fixed execution phases.
  - Static GNNs can be decomposed into aggregation and combination phases.
    - Aggregation phase uses graph structure and neighbor interactions to update the feature vector of the nodes.
    - Combination phase passes the aggregated features through a neural network to compute node embeddings.
  - TGNNs cannot be decomposed into such simplistic phases.

Challenges in Designing a TGNN Accelerator (2)

- Must be able to handle both memory-bound and compute-bound kernels
  - This table shows the operational intensity of various TGNN kernels.
  - **Red** indicates memory-bound kernels, **blue** indicates compute-bound kernels.

	Message aggregator function					Memory updater function					Embedding function				
		Batch size = 1		Batch size = 32			Batch size = 1		Batch size = 32			Batch size = 1		Batch size = 32	
	Type	Wiki	GDELT	Wiki	GDELT	Type	Wiki	GDELT	Wiki	GDELT	Type	Wiki	GDELT	Wiki	GDELT
TGN-attn [1]	id	0.07	0.05	0.10	0.06	GRU	0.50	0.50	15.04	15.63	attn (1 layer)	3.11	2.62	83.03	69.68
TGN-sum [1]	id	0.07	0.05	0.10	0.06	GRU	0.50	0.50	15.04	15.63	sum	0.71	0.48	0.77	0.49
TGAT [2]	-	-	-	-	-	-	-	-	-	-	attn (2 layers)	28.09	26.60	116.97	151.61
JODIE [3]	id	0.07	0.05	0.10	0.06	RNN	0.05	0.05	13.47	14.82	time projection	0.50	0.50	0.97	0.97
tFPGA [4]	id	0.07	0.05	0.10	0.06	GRU	0.50	0.50	15.04	15.56	simple attn	1.03	0.74	1.17	0.78

  - Even the same kernel can be either memory- and compute-bound based on the batch size!
  - TGNN accelerator have to efficiently support both types of workloads.
- Workload balancing is also a critical issue in evolving graphs.
  - Cannot preprocess and statically partition the nodes.

[1] Rossi et al., arXiv, 2020 [2] Xu et al., arXiv, 2020 [3] Kumar et al., SIDKDD, 2019 [4] Zhou et al., IPDPS, 2022.

TGN-PNM: Key Idea

- Near-memory accelerator targeting TGNN workloads.
- Built on 3D-stacked memory (Hybrid Memory Cube).
- Vault-level parallelism:
  - Each vault has a Vault Processing Unit (VPU).
  - VPU = SIMD unit (memory-intensive ops) + systolic array (compute-intensive ops).
- Achieves linear scalability with memory capacity.

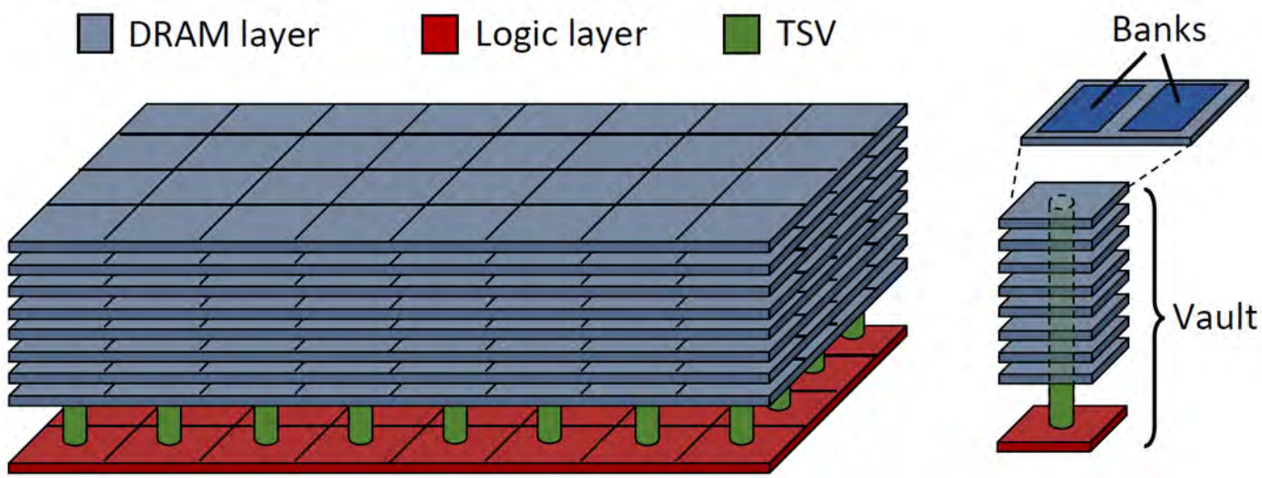
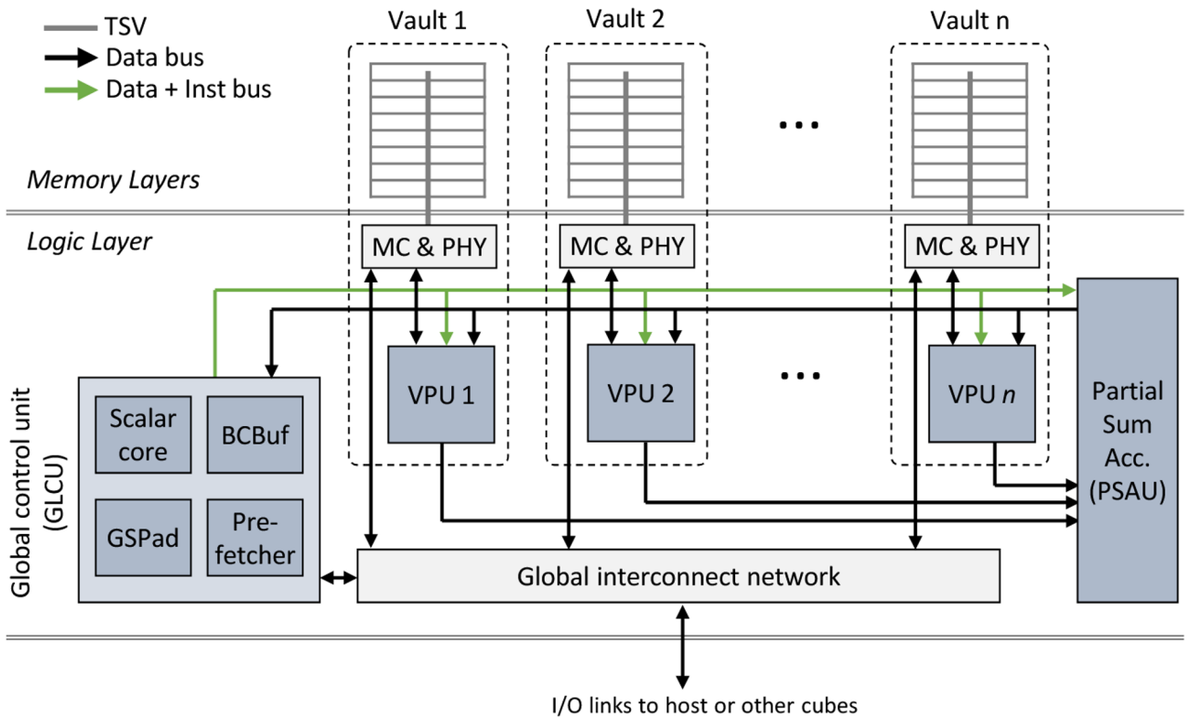
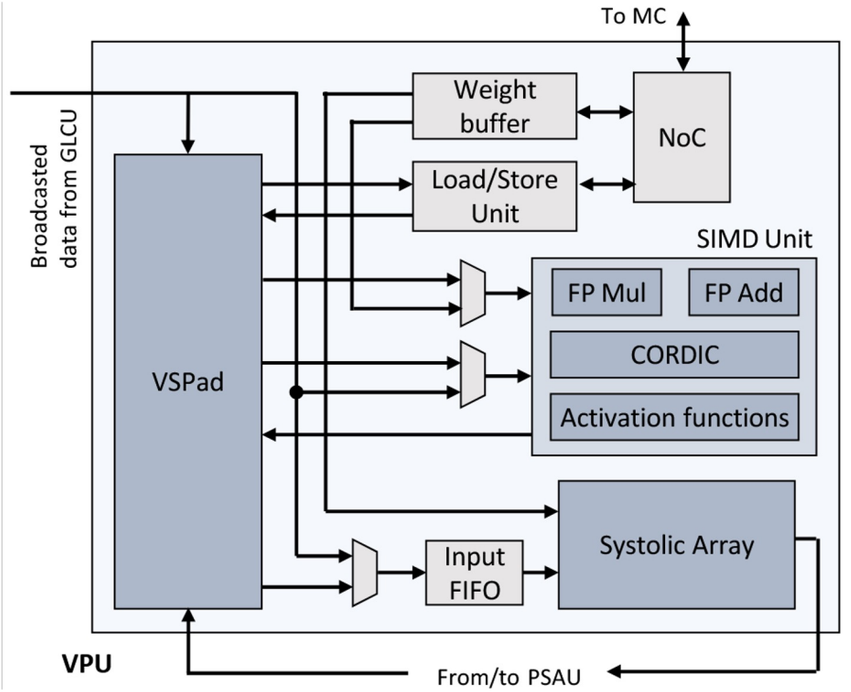


Fig: Hybrid Memory Cube. VPUs are placed in the logic layer.

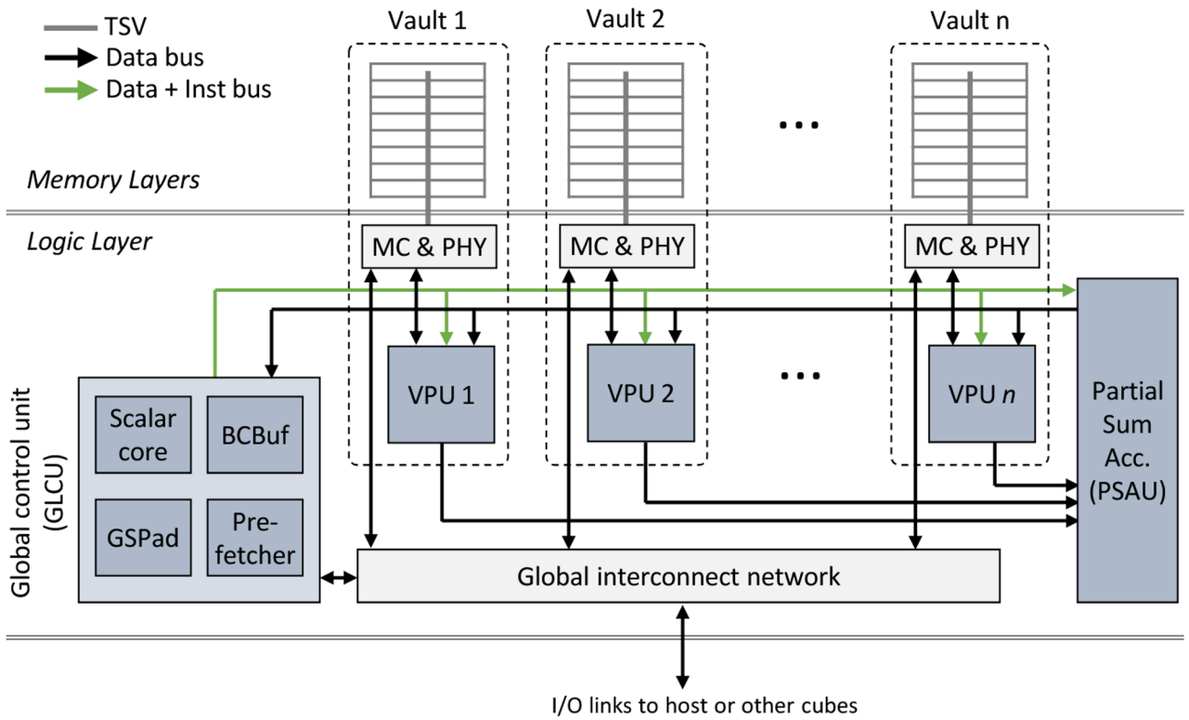
TGN-PNM: Architecture Overview

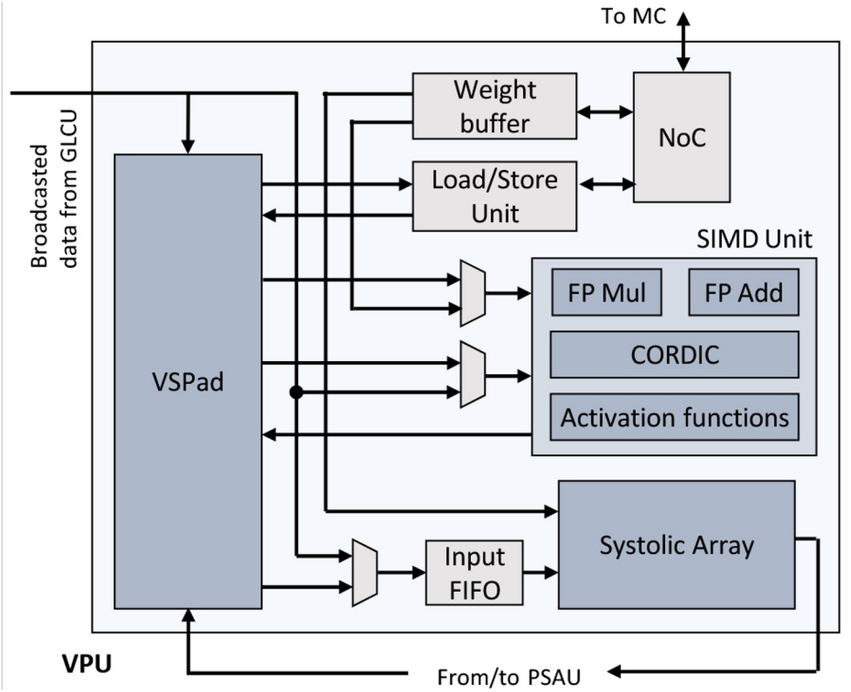




- Main components:
  - Vault Processing Units (VPUs): Core compute.
  - Global Control Unit (GLCU): Instruction broadcast & scalar tasks.
  - Partial-Sum Accumulation Unit (PSAU): Efficient inter-vault reductions.

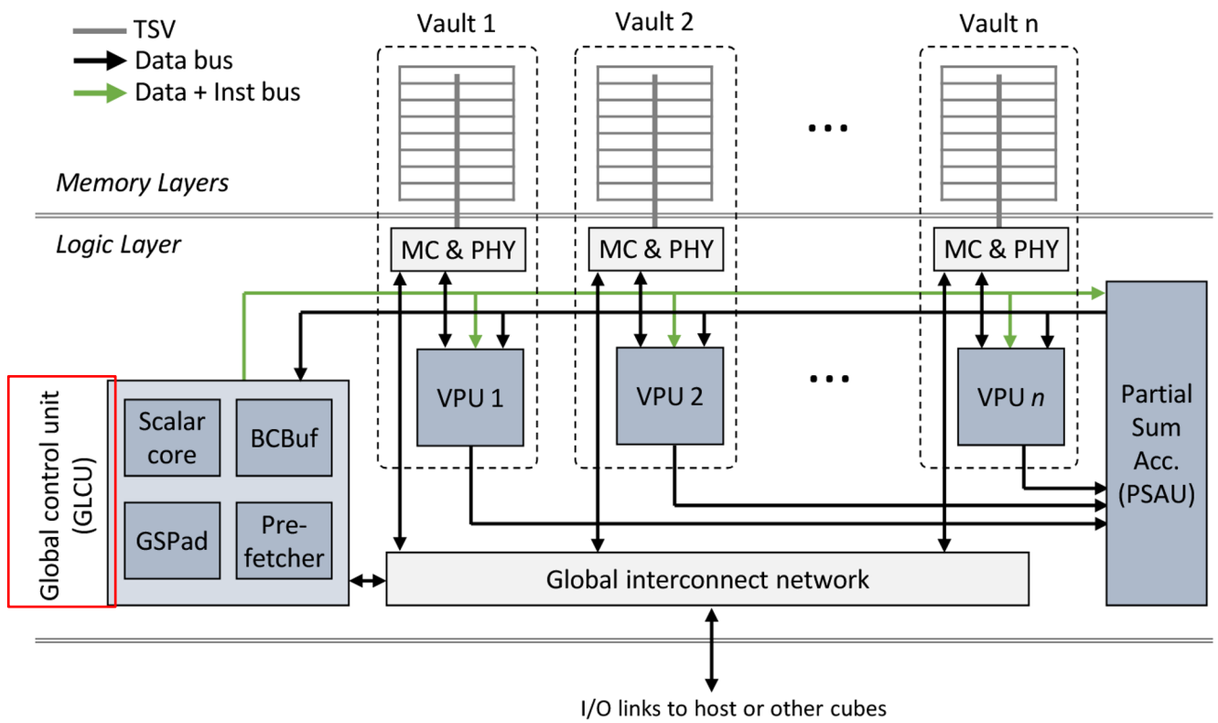
TGN-PNM: Architecture Overview

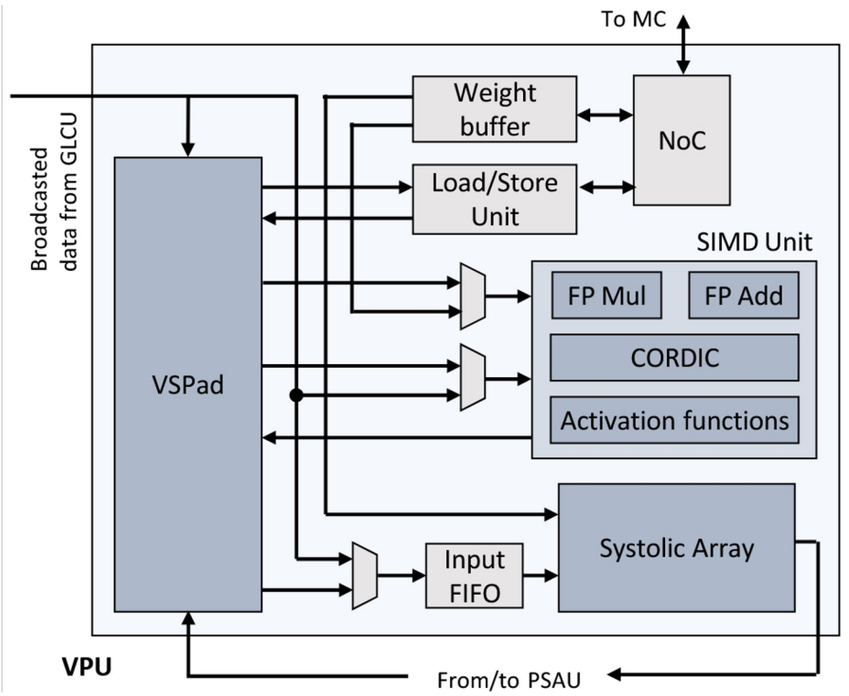




- Vault Processing Units (VPUs):
  - Each vault in the HMC logic layer contains one VPU.
  - Each VPU has a SIMD unit (for memory-bound ops) and a systolic array (for GEMM).
    - Operands for these can come from various sources: scratchpad memory, broadcasted scalar value from GLCU, or cached weights from the weight buffer.
  - All VPUs are operated in lockstep, controlled by the GLCU.

TGN-PNM: Architecture Overview





- Global Control Unit (GLCU):
  - Operates the VPUs by broadcasting instructions.
  - Contains a scalar core for doing complex operations that cannot be done by VPUs.
  - Can broadcast scalar values to all VPUs by writing the values to a broadcast buffer.

TGN-PNM: Workload Balancing

- Naive approach:
  - Partition the graph by nodes, where each vault contains a subset of the nodes.
  - Requires significant inter-vault communication during neighbor aggregation.
- Feature-dimension partitioning:
  - Each vault holds a ***subset of the features*** of ***all nodes***.
  - All element-wise operations becomes localized within each VPU.
  - Inter-vault communication is only required for dot-product reduction operations during matrix-vector and matrix-matrix multiplications.
    - Has a regular pattern and can be handled efficiently by the partial-sum acc unit.
  - Does not need to duplicate weights across all the vaults.
  - Enables lockstep operations of the VPUs.
  - However, does not work well if the feature dimension is small as we cannot utilize all SIMD and systolic array lanes!
- Hybrid partitioning:
  - For small feature dimension, sub-partitions the lanes and use duplicate weights.

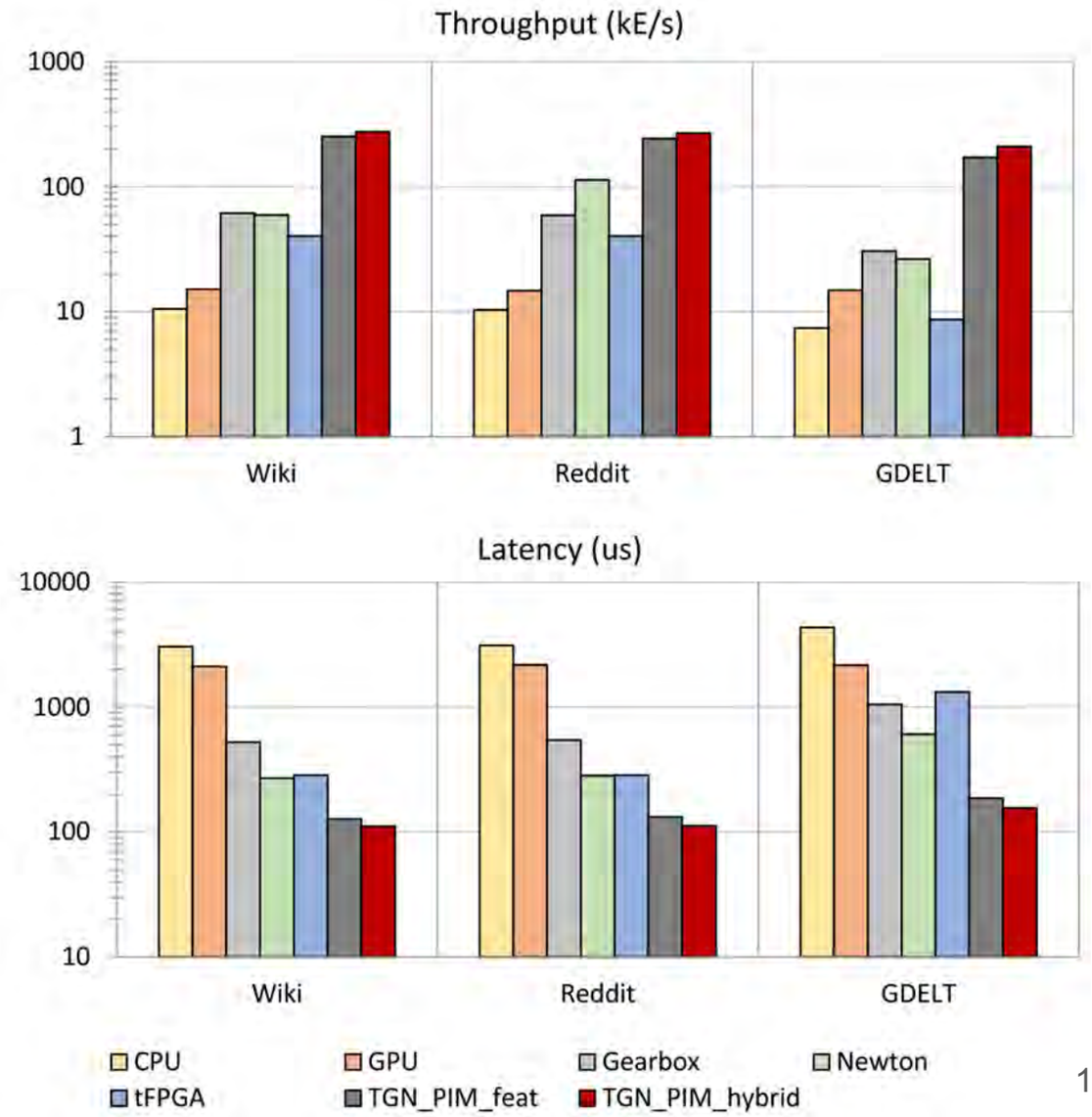
Evaluation Setup

- Benchmarks: TGN-attn & TGN-sum model architectures [1]
  - TGN-attn: Single layer multi-head attention for neighbor aggregation
    - Memory + compute intensive
  - TGN-sum: Simple average for neighbor aggregation
    - Memory intensive
- Datasets: Wikipedia, Reddit, GDELT
- Two variants of our proposed architecture:
  - TGN_PIM_feat uses feature-dimension partitioning scheme
  - TGN_PIM_hybrid uses hybrid partitioning scheme
- Comparisons against:
  - CPU (AMD EPYC 7742 64-core)
  - GPU (NVIDIA A100)
  - Gearbox (subarray-level PIM) [2]
  - Newton (bank-level PIM) [3]
  - tFPGA (FPGA-based TGNN accelerator) [4]

[1] Rossi et al., arXiv, 2020. [2] Lenjani et al., ISCA, 2022. [3] He et al., MICRO, 2020. [4] Zhou et al., IPDPS, 2022.

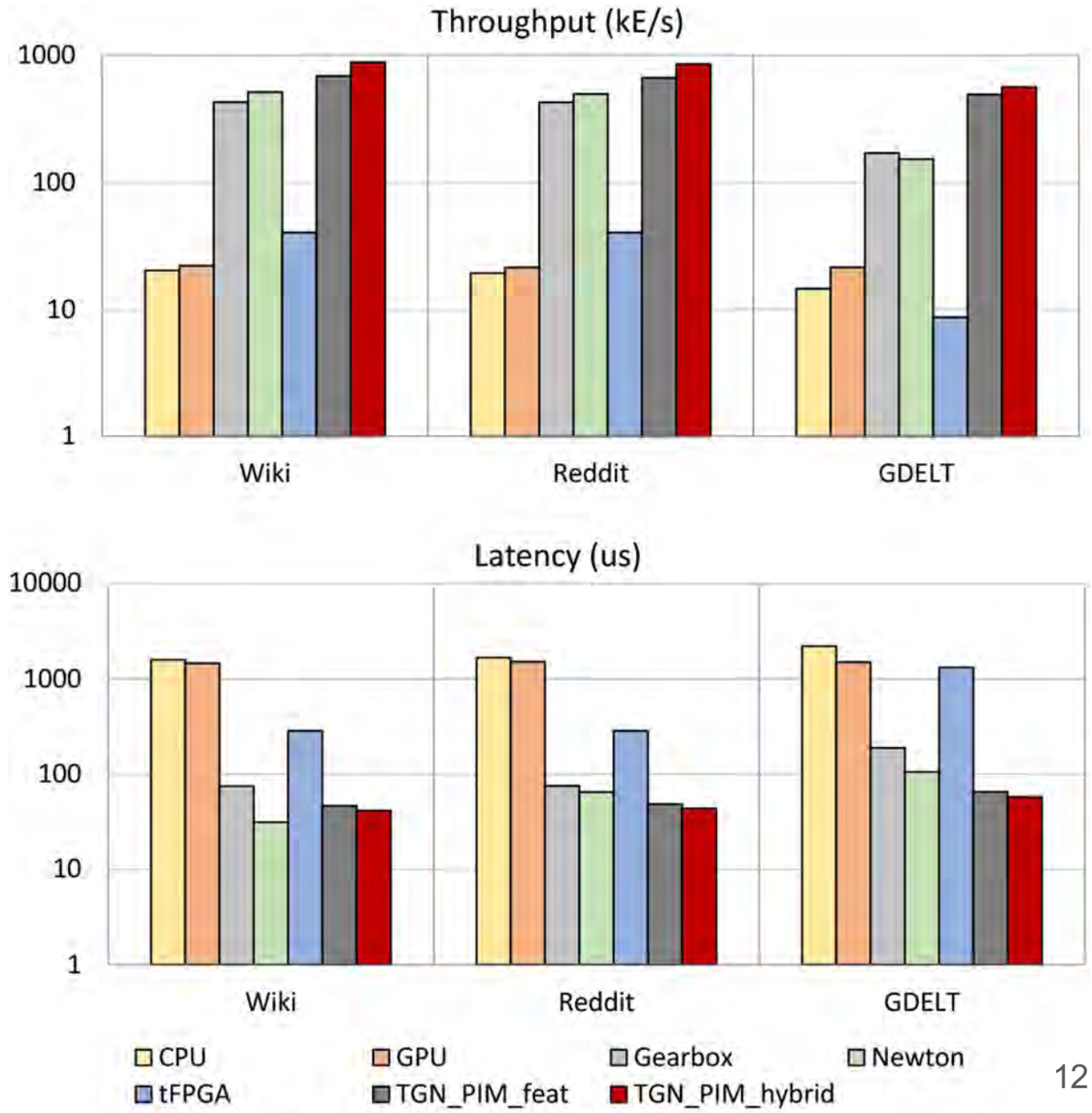
Results – Throughput & Latency - TGN-attn

- On TGN-attn benchmark, TGN_PIM_hybrid provides the best throughput:
  - 26.8× over CPU
  - 16.7× over GPU
  - 5.2× over Gearbox
  - 4.4× over Newton
  - 10.4× over tFPGA
  - 1.14x over TGN_PIM_feat
- Significant reductions in batch processing latency across datasets.

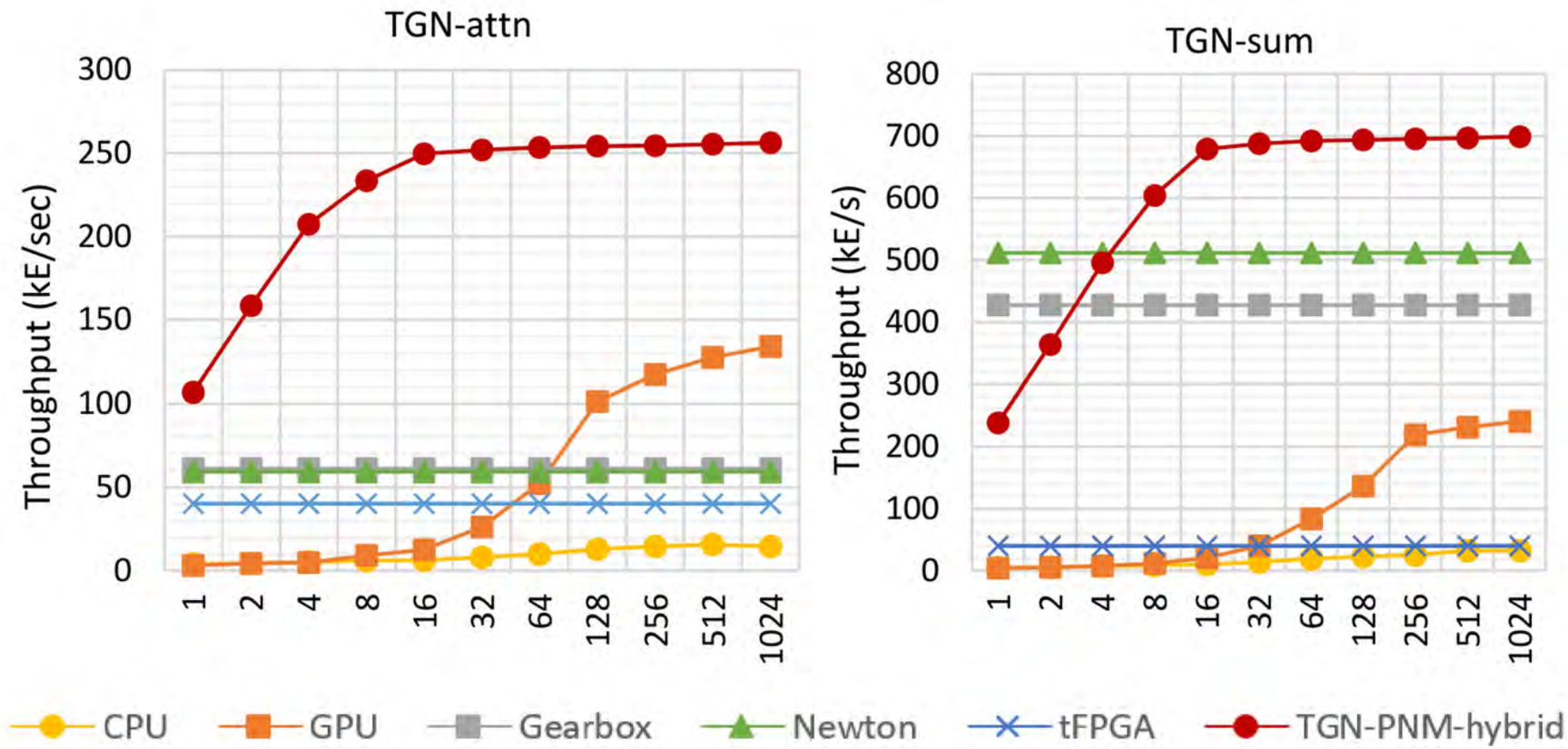


Results – Throughput & Latency - TGN-sum

- On TGN-sum benchmark, TGN_PIM_hybrid provides the best throughput:
  - 42.1x over CPU
  - 34.8x over GPU
  - 2.4x over Gearbox
  - 2.2× over Newton
  - 31.1× over tFPGA
  - 1.23x over TGN_PIM_feat
- Gearbox and Newton performed fairly well due to the benchmark being memory bound



Impact of Batch Size



- TGN-PNM throughput improves with batch size up to SIMD width (16).
  - At this point, systolic arrays leverages maximum reuse.
- Batching does not impact Gearbox and Newton.
  - No reuse because GEMM implemented as multiple independent GEMV ops.

Area Estimation

Component	Count	Per unit area (mm^2)	Total area (mm^2)
Scalar core (ARM Cortex A-35)	1	0.68	0.68
Partial Sum Acc. Unit	1	2.25	2.25
Global NoC	1	1.43	1.43
GSPad	1	1.03	1.03
Memory controller and DDR PHY	32	0.18	5.90
Systolic array	32	0.25	8.08
SIMD: FP add & mul	32	0.02	0.51
SIMD: CORDIC	32	0.03	0.96
SIMD: Activation	32	0.15	4.68
VSPad	32	0.32	10.31
Total:			**35.83**

On SAED 14nm node

- Processing elements and control logics: by RTL synthesis
- SRAM buffers and scratchpad memories: using CACTI-3DD
- Memory controller and interconnect: using McPAT

Total estimated area is 35.83mm^2, which is 53% of the total logic-die area of 68mm^2 HMC stack.

Conclusions

- First near-memory accelerator for TGNN inference.
- Efficient handling of both memory- and compute-bound kernels.
- Feature-dimension partitioning ensures balanced workload.
- Outperforms CPU, GPU, FPGA, and other general purpose PIM architectures.
- Scalable and flexible for future TGNN variations.

Thank you!

Questions?