GOPIM: GCN-Oriented Pipeline Optimization for PIM Accelerators

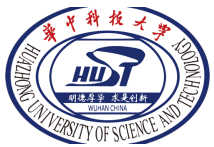
Siling Yang, Shuibing He, Wenjiong Wang,Yanlong Yin, Tong Wu,
Weijian Chen, Xuechen Zhang, Xian-He Sun, Dan Feng

浙江大学
Zhejiang University

ZHEJIANG LAB
之江实验室

WASHINGTON STATE UNIVERSITY

ILLINOIS TECH



HPCA 2025

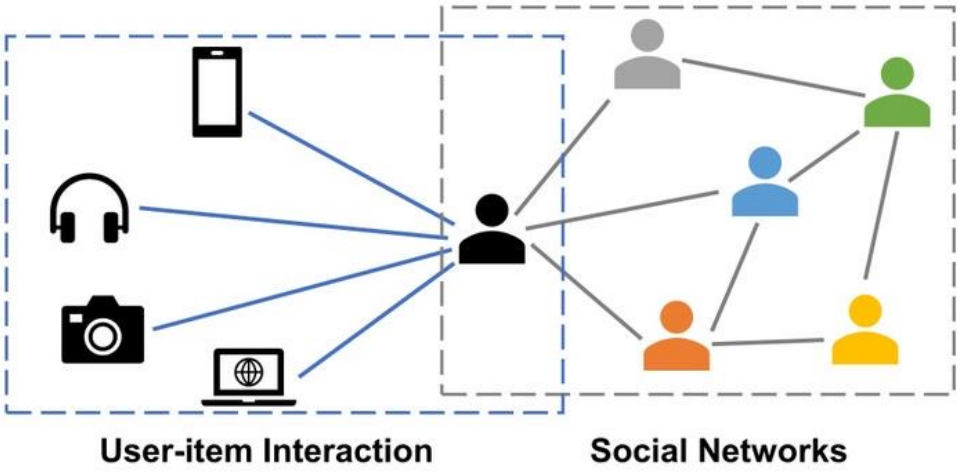
Outline

- Background & Motivation
- Initial Ideas and Challenges
- Design of GOPIM
- Evaluation
- Conclusion

Graph Convolutional Network

- **Graph convolutional networks (GCN) are widely used in node classification, link prediction and recommendation.**

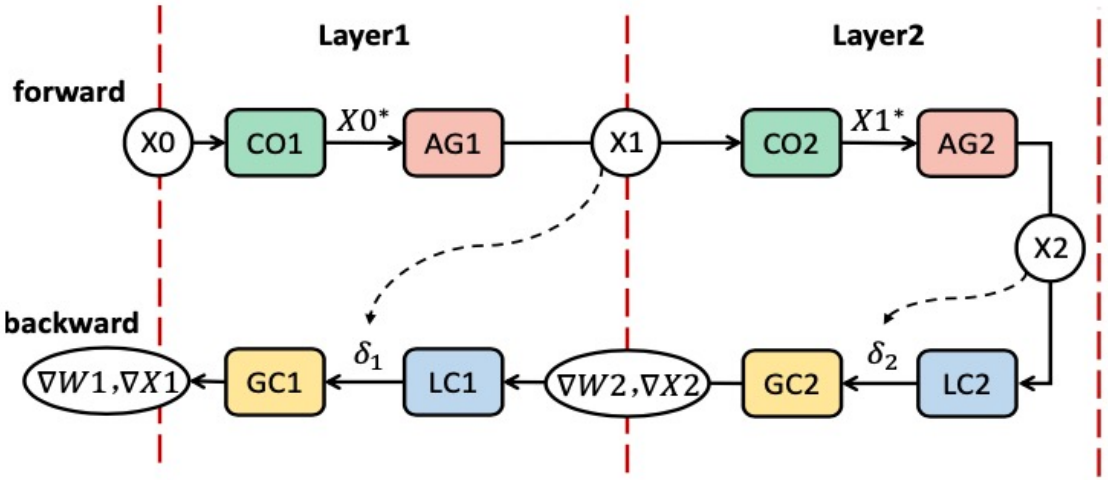
Node classification **Link prediction** **recommendation**



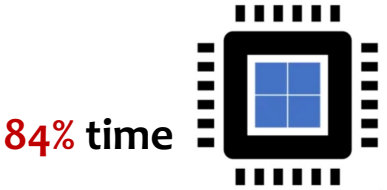
- ***Aggregation:***
  - ✓ **Integrate self and neighbor information**
- ***Combination :***
  - ✓ **Learn and refine integrated information**

Graph Convolutional Network

GCN training

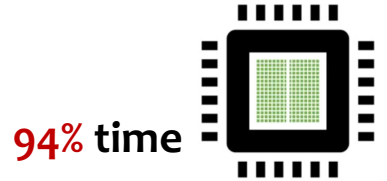


Forward： Combination (CO), Aggregation (AG);
Backward： Error computation (LC), gradient computation (GC)



84% time

CPU platforms



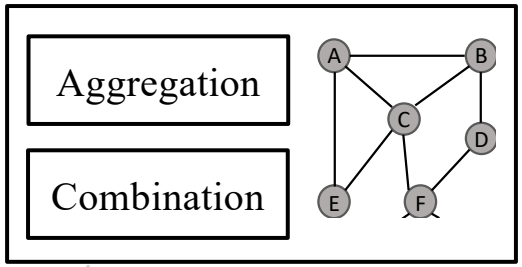
94% time

GPU platforms

There are many irregular memory accesses in Aggregation stages, causing a performance bottleneck in the system.

Processing-in-Memory

Traditional

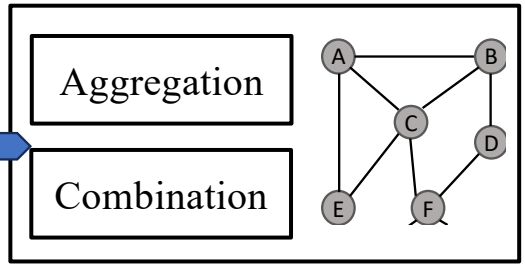


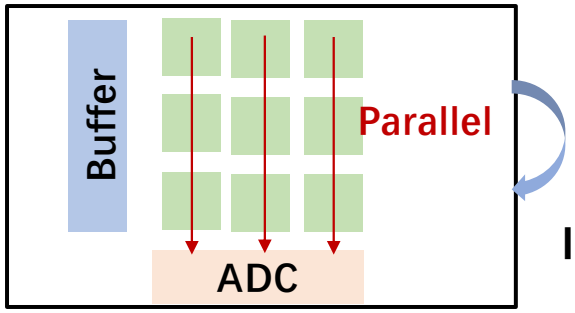
Numerous irregular accesses

MEM

MVM

Processing-in-memory (PIM)





In-situ computation

Computation

High parallel execution of MVM
Low hardware overhead

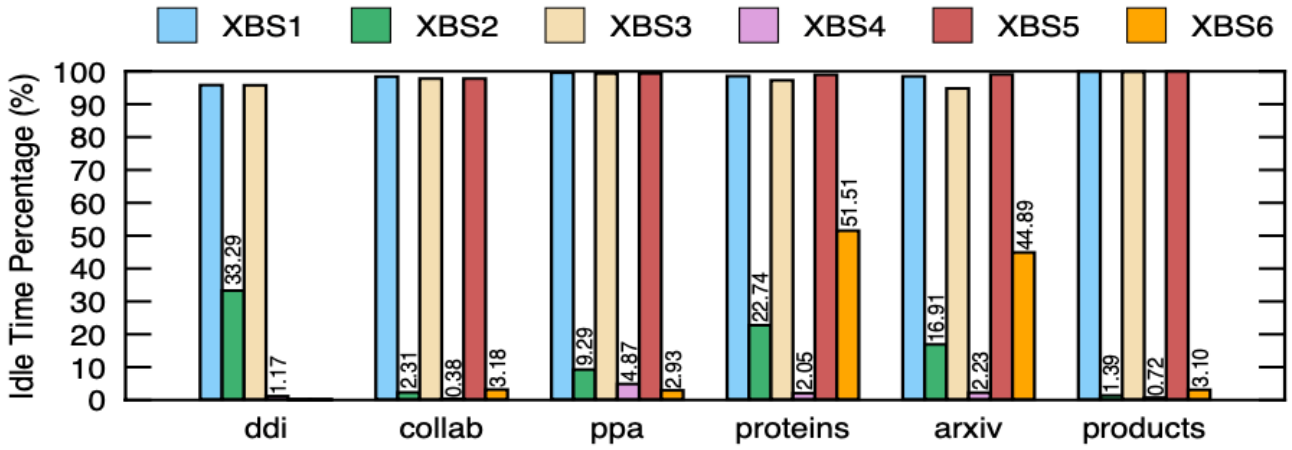
Memory access

In-situ computation on memory unit
Low memory access overhead

PIM shows superior performance over traditional architectures for GCN .

Motivation 1:

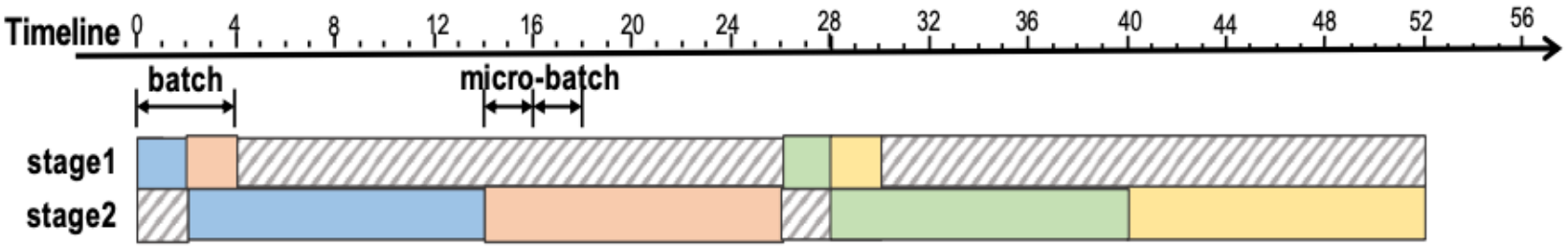
- **Motivation 1: Existing ReRAM-based GCN accelerators do not fully resolve the under-utilization issues of ReRAM crossbars.**



SlimGNN* on GCN model with six datasets

Motivation 1:

- The idleness are mainly caused by two reasons.
  - First, data dependencies exist among stages.
  - Second, different stages have distinct computation patterns (up to 888x).

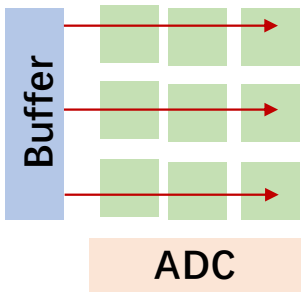


(a) Baseline. Unused crossbar: 3. Execution time units: 52

Motivation 2 :

- **Motivation 2: vertex updating operations (writing mapped vertices onto crossbars) can be time-consuming.**

Writing mapped vertices onto crossbars



52% of the execution time of AG1 and AG2 on the *ppa* dataset

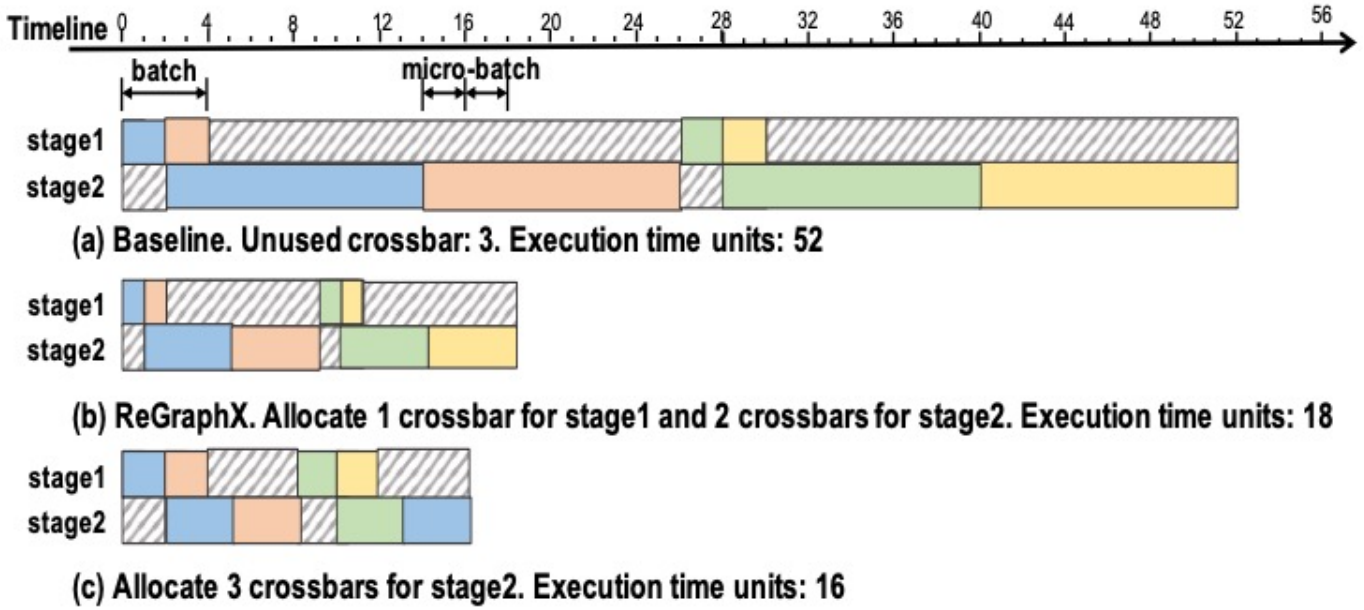
Our Initial Idea:

Use unoccupied crossbar resources as replicas to shorten the execution times of longer stages, thus streamlining the overall pipeline.

Selectively update only a portion of vertices on ReRAM-based PIM architectures to optimize performance.

Challenge 1:

- **Challenge 1: It is difficult to determine how many replicas to set, since the execution times vary at different stages.**

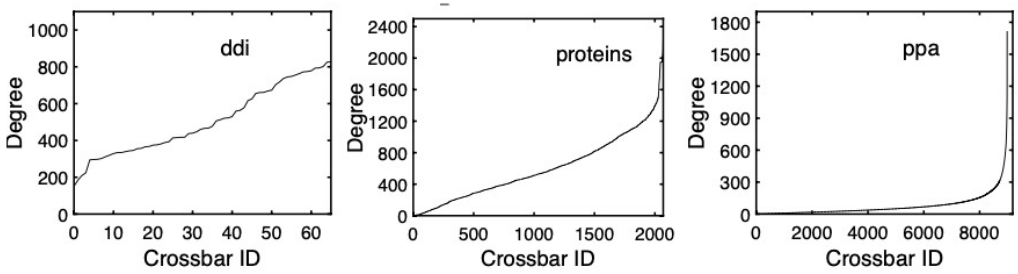


Challenge 2:

- **Challenge 2: Existing mapping strategies for ReRAM-based GCN accelerators, such as ReGraphX and SlimGNN, may negate the latency benefits of selective updating.**

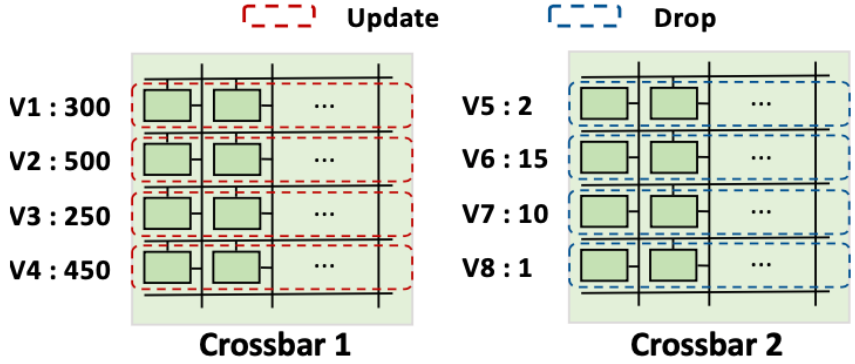
+ Selective updating vertices

Vertex index-based mapping strategies ⇒ **Original Selective Updating (OSU)**



151.8 ~ 827.4 1.6 ~ 2266.8 1 ~ 1716.91

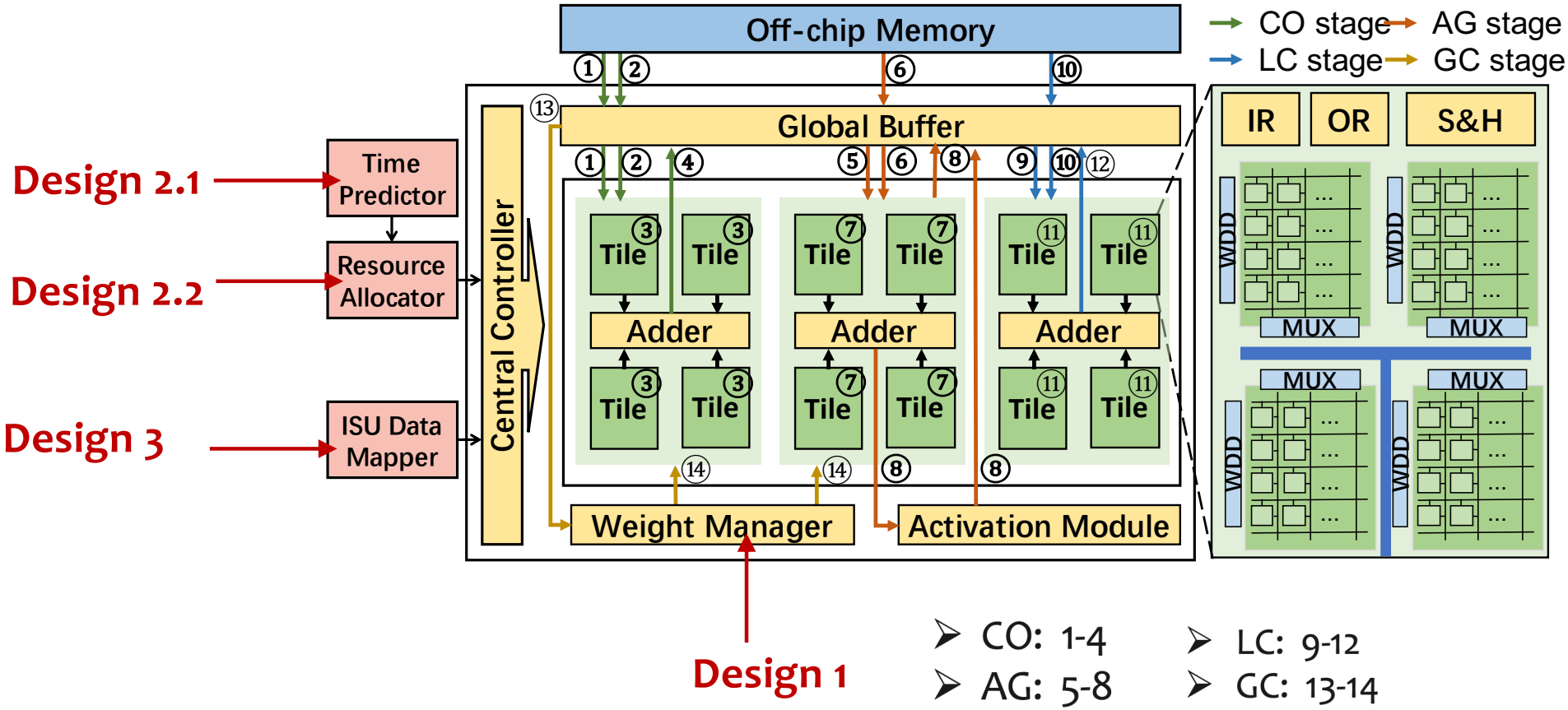
biased degree distribution



Without sparsification: 4 cycles
OSU with sparsification: 4 cycles

Design 1: Architecture and dataflow

ReRAM-SRAM architecture for GCN accelerator

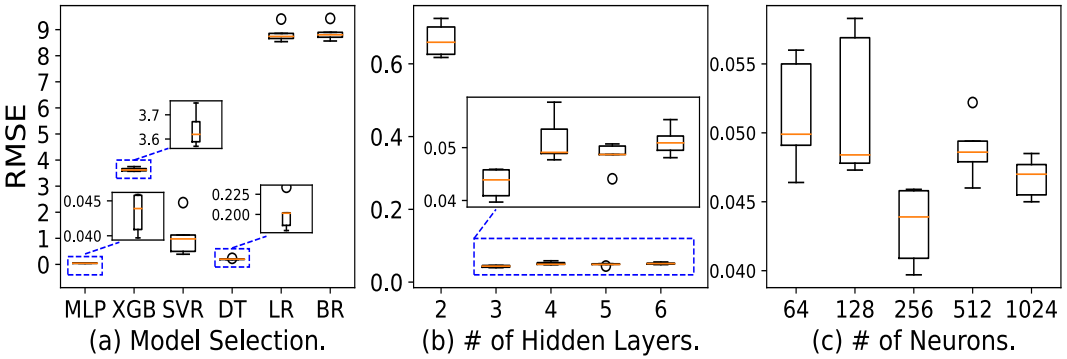


- CO: 1-4
- AG: 5-8
- LC: 9-12
- GC: 13-14

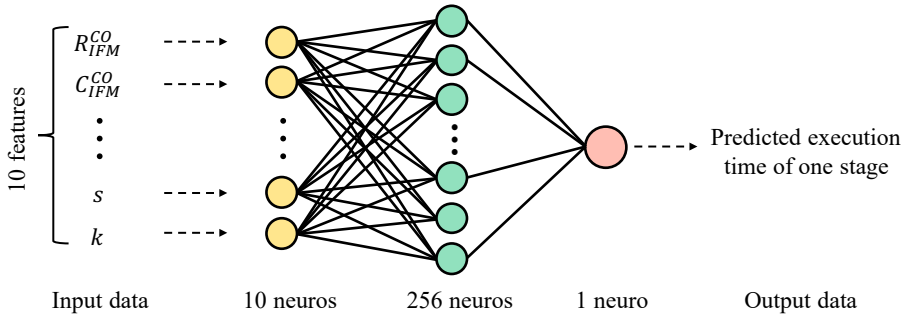
Design 2.1: ML-based Execution Time Prediction

Key idea: (1) Use the Multilayer perceptrons (MLP) model to predict stage execution times on a crossbar-based PIM accelerator, (2) then greedily allocate replicas based on these times.

(1) ML-based Execution Time Prediction



The model decision process

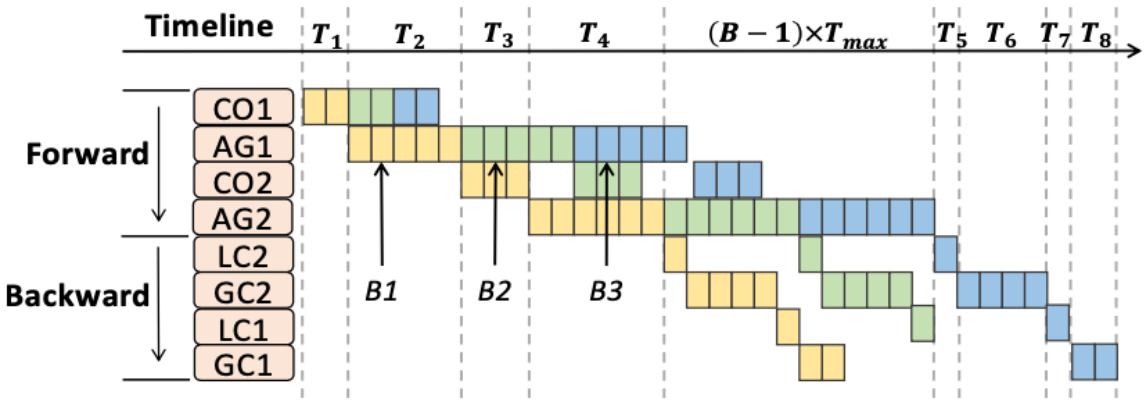


The model structure

Design 2.2: Max-Heap Based Resource Allocator

Key idea： Use an MLP model to predict stage execution times on a crossbar-based PIM accelerator, then greedily allocate replicas based on these times.

(2) Max-Heap Based Resource Allocator



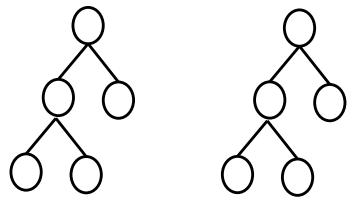
$$T_i^j(start) \geq T_i^{j-1}(end)$$

$$T_i^j(start) \geq T_{i-1}^j(end)$$

$$T_{max}^j = \max(T_i^j), i = 1, 2, 3..., 4L$$

$$T_A = \sum_{i=1}^{4L}(T_i^j) + (B-1) \times T_{max}^j$$

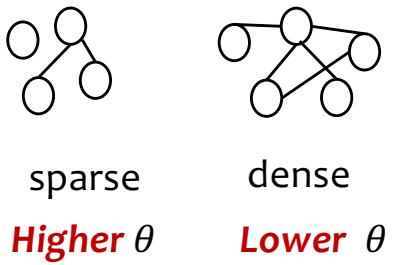
Adjustable value Execution time



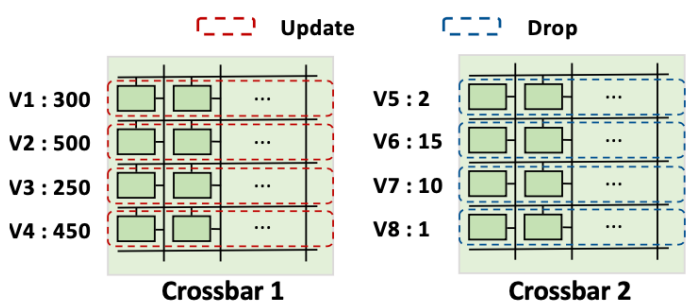
Design 3: Interleaved mapping with adaptive selective update method

Key idea: Update important vertices every epoch, less important vertices every 20 epochs. Use interleaved mapping to balance writes across crossbars.

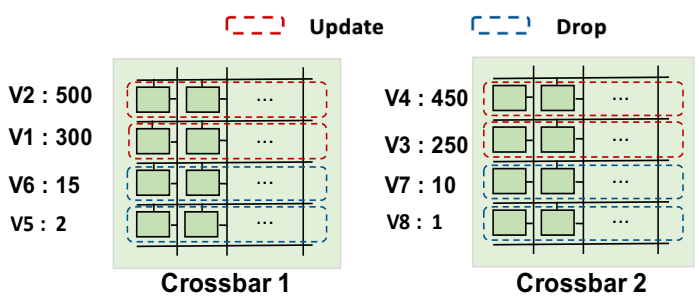
Adaptive selective vertex updating



Interleaved mapping with selective update



Original (OSU)



Ours (ISU)

Evaluation Setup

- System configuration

PIM-based simulator, NeuroSim

- Workloads and datasets

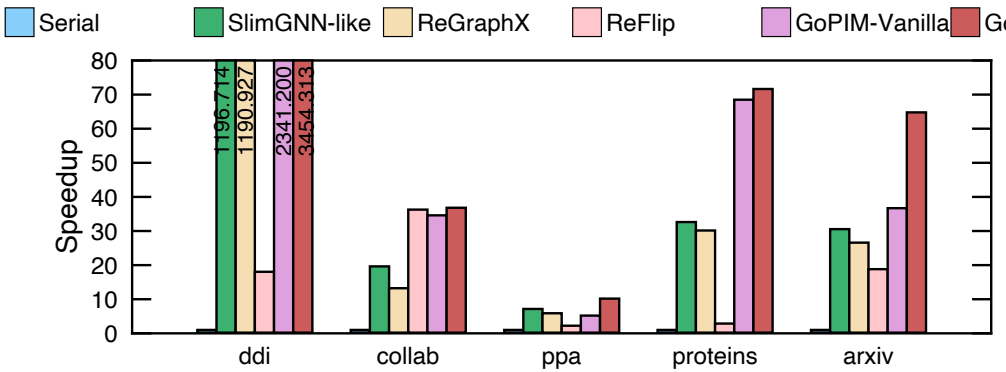
Datasets	Link prediction: Ddi, collab, ppa, Node prediction:proteins, arxiv, products, Cora
Models	Six GCN models (256-256-256, 128-256-256, 58-256-256, 8-256-112, 128-256-40, …)

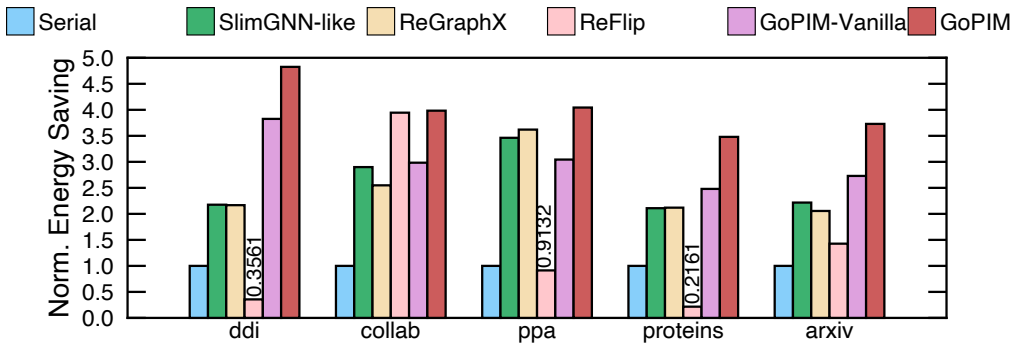
- Compared systems

Serial	sequential execution without pipeline and graph sparsification;
SlimGNN-like [TCAD22]	SlimGNN without weight pruning;
ReGraphX [DATE21]	a fixed resource allocation ratio and it discards graph sparsification
ReFlip [hpca22]	adopts replicas only in combination phases and without sparsification
GOPIM-Vanilla	GOPIM without using ISU
GOPIM	ours

Overall Performance and Energy Saving

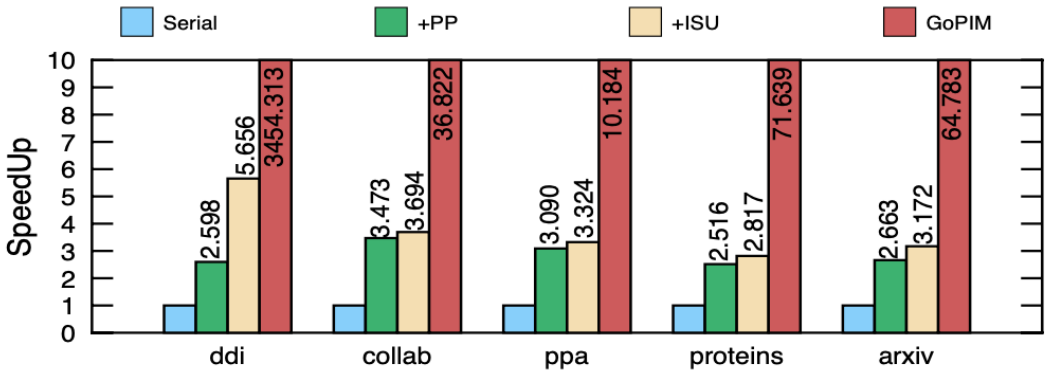
GOPIM gains the largest speedup and energy saving for all datasets.



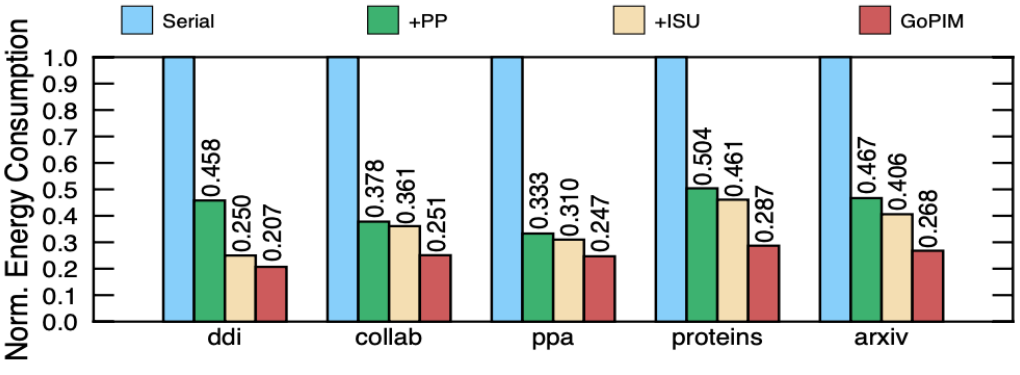


GOPIM achieves 727.6×, 2.1x, 2.4×, 45.1×, and 1.5× on average, compared to Serial, SlimGNN-like, ReGraphX, ReFlip, and GOPIM-Vanilla, respectively.

Impact of Individual techniques



(a) Speedup.



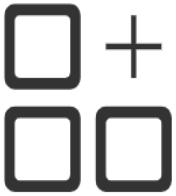
(b) Energy Comsumption.

+PP, +ISU, and GOPIM deliver up to 62%, 75%, and 79% energy reduction, respectively, across all datasets.

Additional Results in Paper

Accuracy analysis

Sensitivity analysis



Scalability analysis

Overhead analysis

Thanks & QA

GOPIM:GCN-Oriented Pipeline Optimization for PIM Accelerators

浙江大学
Zhejiang University

ZHEJIANG LAB
之江实验室

WASHINGTON STATE UNIVERSITY

ILLINOIS TECH

Contact Email: slingzjunet@zju.edu.cn

ISCS Lab: https://shuibing9420.github.io