PIMMiner: A High-performance PIM Architecture-aware Graph Mining Framework

Presenter: Jiya Su   Advisor: Rujia Wang

Computer Science of Illinois Institute of Technology

MICRO-55

ILLINOIS INSTITUTE OF TECHNOLOGY

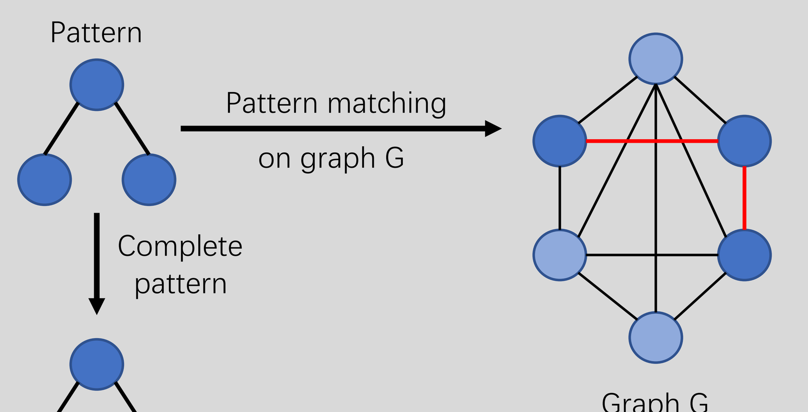
Abstract

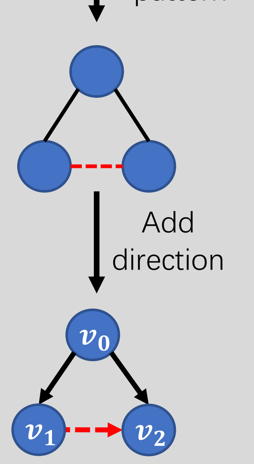
We propose PIMMiner, a high-performance PIM architecture graph mining framework.

- We first identify that current PIM architecture cannot be fully utilized by graph mining applications.
- Next, we propose a set of optimizations that enhance the locality, and internal bandwidth utilization and reduce remote bank accesses and load imbalance through cohesive algorithm and architecture co-designs.
- We compare PIMMiner with several state-of-the-art graph mining frameworks and show that PIMMiner is able to outperform all of them significantly.

Background

- Graph pattern mining (GPMI) needs to find all subgraphs with different patterns that meet the application requirements.
- GPMI applications are considered as a new class of data-intensive applications that generate massive irregular computation workloads and memory accesses, which degrade the performance significantly.





AutoMine Algorithm:

$for\ v_0 \in V$
$for\ v_1 \in N(v_0)\ and\ v_1 \neq v_0$
$for\ v_2 \in N(v_0) - N(v_1)$
$and\ v_2 < v_1\ and\ v_2 \neq v_0$
(v_0, v_1, v_2) *is an embedding*
$counter \mathrel{+}= 1$

- Processing-in-Memory (PIM) integrates processing units inside the memory to reduce the overhead of frequent data movement and achieve high-performance and energy-efficient computation.
- Samsung has recently started manufacturing HBM-PIM chips. The HBM-PIM incorporates PIM cores inside of memory banks. There are three ways for a PIM unit to access memory: (1) near-core bank access, (2) intra-channel bank access; (3) inter-channel remote bank access.

Motivation

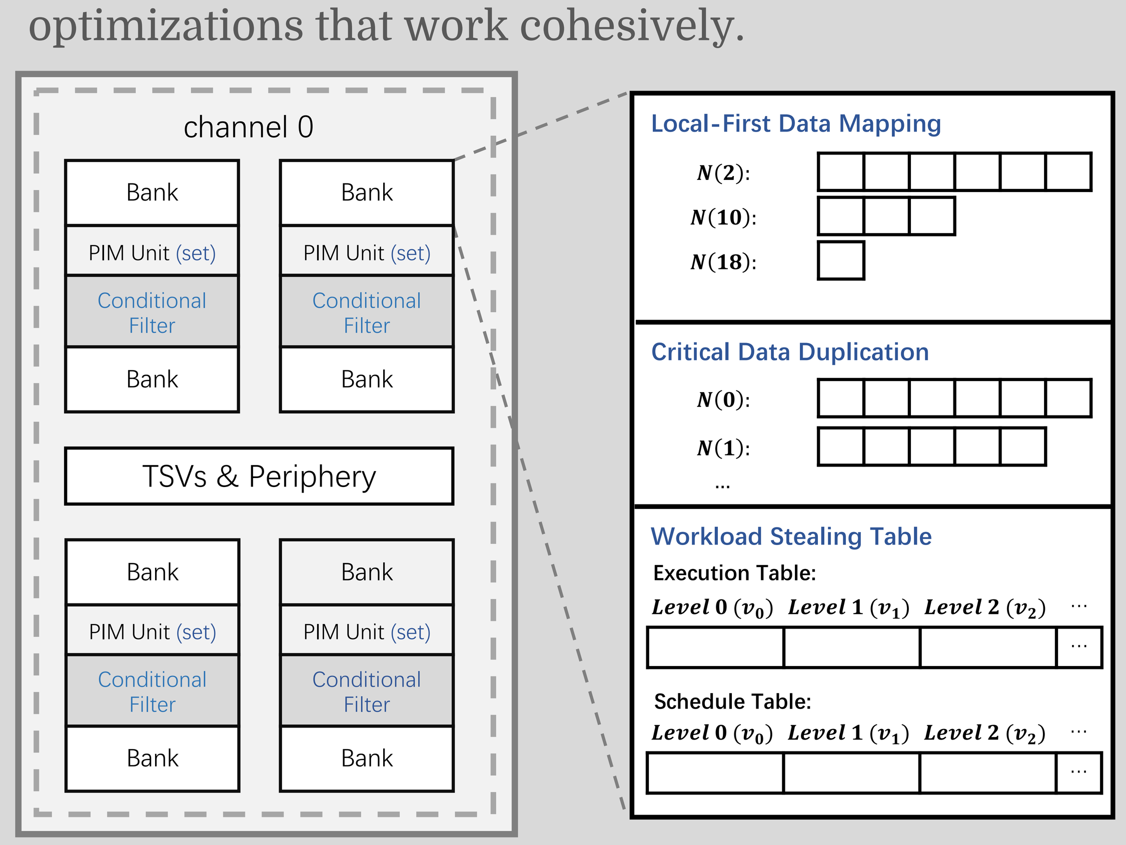
- Directly offload the GPMI execution kernel to PIM cannot achieve desired performance. We observe **high load imbalance** and lots of **inter-channel remote bank accesses**.

TABLE I: Performance between 96-threads CPU (L3 cache) and 128-core PIM (L1 cache) in 4-CC.

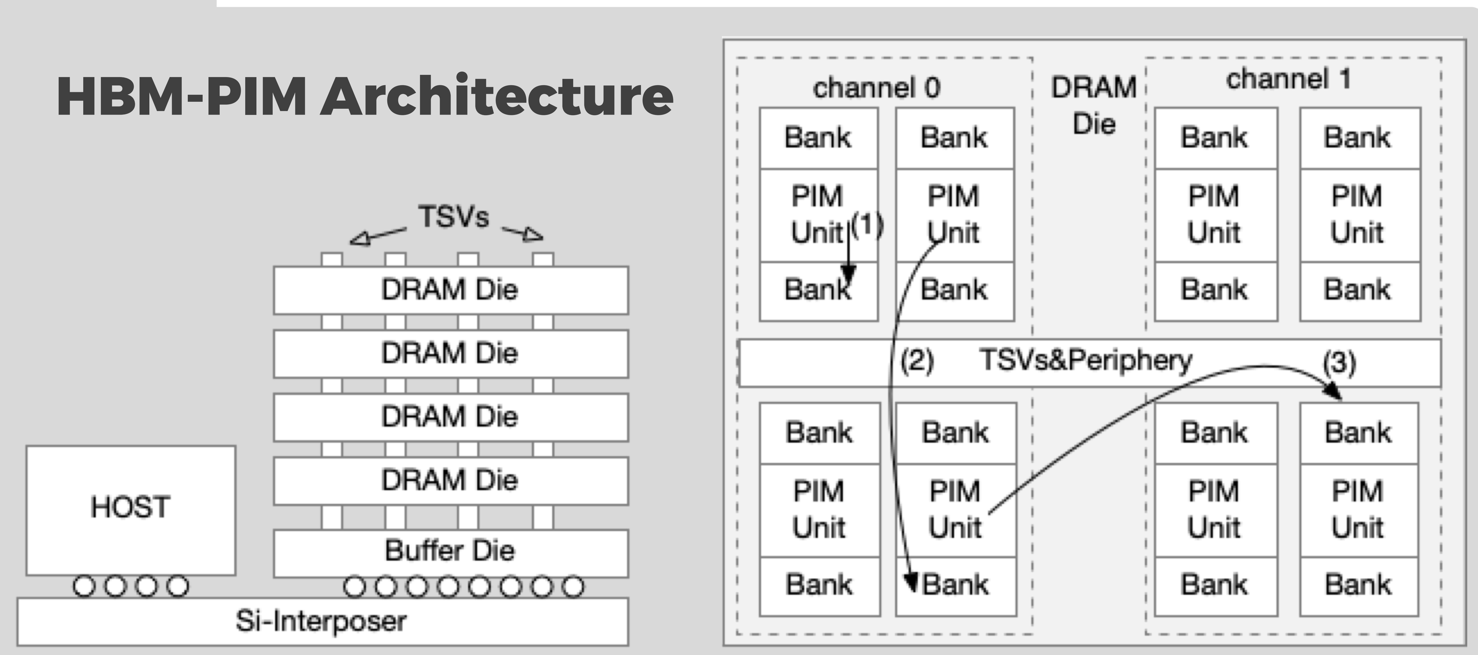
Graph	Processing Unite	Execution Time (s)	Average Time (s)	Exe/Avg	Execution Speedup	Average Speedup
CI	CPU PIM	3.80E-05 4.13E-05	1.65E-05 3.57E-05	2.30 1.16	**0.92x**	**0.46x**
PP	CPU PIM	7.07E-04 1.71E-04	3.37E-04 1.56E-04	2.10 1.09	4.15x	2.16x
AS	CPU PIM	1.04E-02 4.05E-03	7.04E-03 2.41E-03	1.48 1.68	2.58x	2.92x
MI	CPU PIM	1.64E-01 4.22E-01	8.48E-02 1.66E-01	1.93 2.53	**0.39x**	**0.51x**
YT	CPU PIM	1.49E-01 1.83	9.18E-02 3.78E-01	1.63 4.84	**0.08x**	**0.24x**
PA	CPU PIM	1.83E-01 6.89E-02	1.37E-01 5.40E-02	1.34 1.28	2.66x	2.53x
LJ	CPU PIM	3.09 103.13	2.59 4.63	1.19 22.29	**0.03x**	**0.56x**

PIMMiner Framework Overview

PIMMiner has five lightweight architectural optimizations that work cohesively.



HBM-PIM Architecture

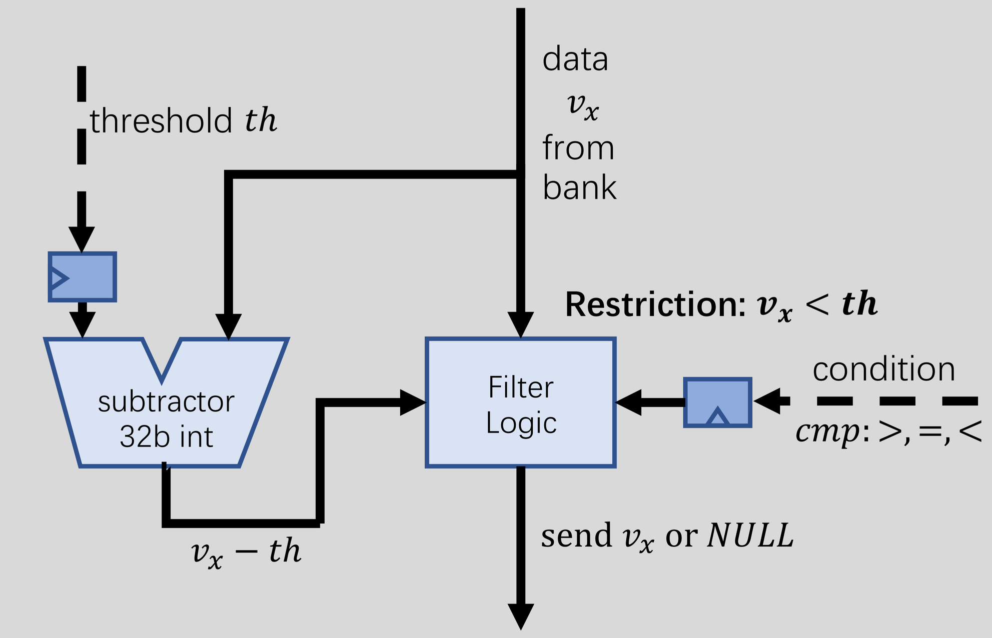


(a) HBM-PIM architecture overview

(b) HBM-DRAM Die and PIM unit memory accesses

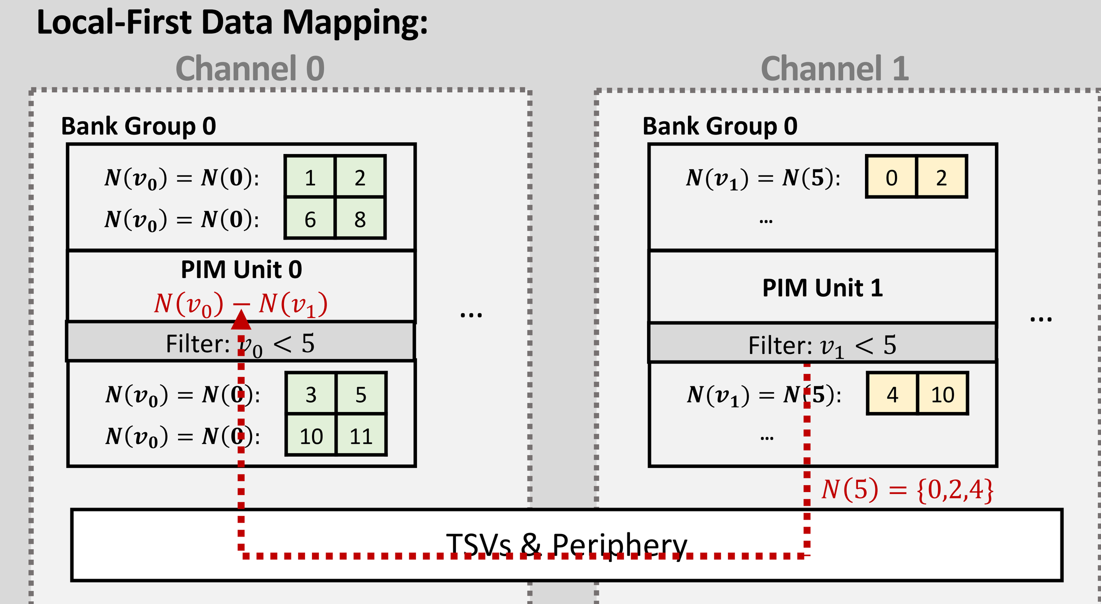
PIMMiner Framework Design

- **Conditional Access Filter (Filter)**: PIMMiner adds lightweight hardware dedicated to filter unnecessary data from memory. According to AutoMine, PIM units only need the nodes $v_2 < v_1$.



- **Local-First Data Mapping (Remap):** We propose a new address mapping method to exploit low-latency and high-bandwidth local memory bank accesses for PIM units. The new mapping maps the data in the same neighbor list to the same bank group.

Local-First Data Mapping:



- **Critical Data Duplication (Duplication):** To further reduce the remote memory accesses, PIMMiner stores critical data in the unused memory space in the memory banks.
- **Workload Scheduler with Stealing (Stealing)**: PIMMiner achieves a PIM-aware workload scheduler by maintaining the execution tables and schedule tables on each PIM unit to address the load imbalance issues.
- **Working with specialized GPMI accelerator (Set)**: PIMMiner can achieve even higher performance with integration of specialized GPMI accelerator in PIM units, such as the accelerator for intersection set operations.

Evaluation Results

- Overall, by enabling all optimizations, PIMMiner achieves 15.91x average speedup and 137.32x maximum speedup over the baseline PIM.
- The performance improvement from PIMMiner optimizations:
  - Filter: 2.01x average, 17.57x maximum speedup
  - Remap: 1.39x average, 2.74x maximum speedup
  - Duplication: 1.84x average, 3.05x maximum speedup
  - Stealing: 3.01x average, 26.87x maximum speedup
  - Set: 1.49x average, 2.26x maximum speedup

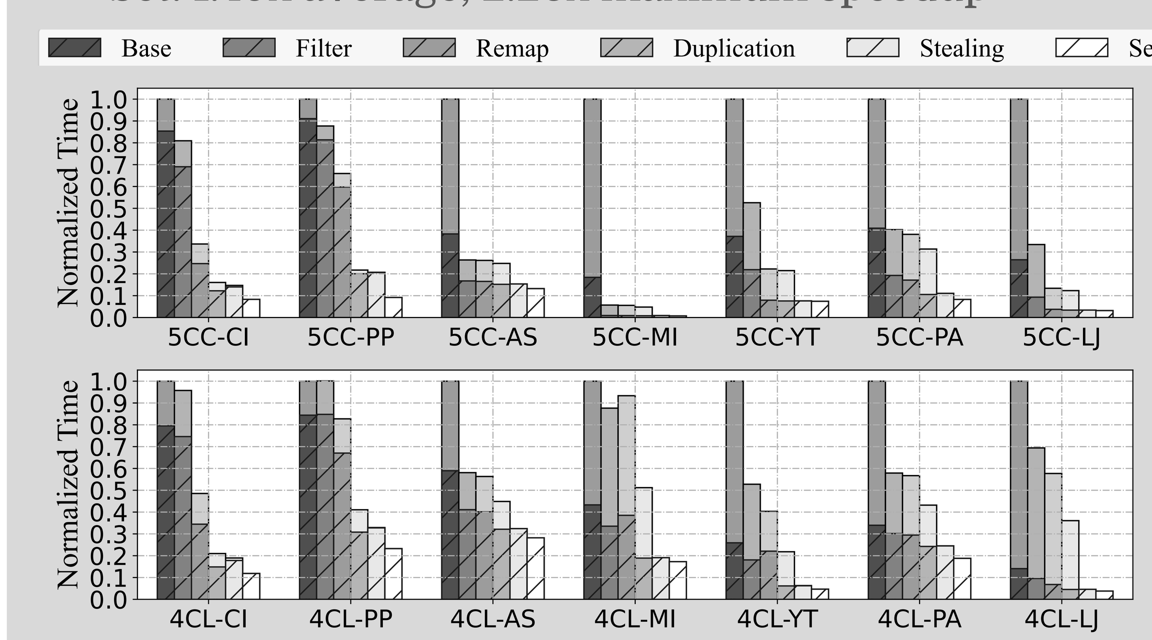


Fig1: Performance of PIMMiner with the effectiveness of proposed optimizations. In each bar, we show the average time across cores (the solid line) and the total execution time(top of bar).

- Compare with other GPMI software systems:
  - GraphPi [1]: 344x average speedup
  - AutoMine [2]: 109x average speedup
- Compare with the hardware GPMI accelerators:
  - Gramer [3]: 696x average speedup
  - FlexMiner [4]: 5.9x average speedup
  - DIMMining [5]: 1.3x average speedup
  - NDMiner [6]: 37x average speedup

References

[1] T. Shi *et al.*, "Graphpi: High performance graph pattern matching through effective redundancy elimination," in *SC, 20.*
[2] D. Mawhirter *et al*, "Automine: harmonizing high-level abstraction and high performance for graph mining," in *SOSP, 20.*
[3] P. Yao *et al.*, "A locality-aware energy-efficient accelerator for graph mining applications," in *MICRO, 20.*
[4] X. Chen *et al.*, "Flexminer: a pattern-aware accelerator for graph pattern mining," in *ISCA, 21.*
[5] G. Dai *et al.*, "Dimmining: pruning-efficient and parallel graph mining on near-memory-computing," in *ISCA, 22.*
[6] N. Talati *et al.*, "Ndminer: accelerating graph pattern mining using near data processing." in *ISCA, 22.*

Acknowledgement

This work is collaboratively done with Dr. Peng Jiang at University of Iowa. Our research is generously supported by the National Science Foundation under grant CCF-2029014 and CCF-2028825.