AceMiner: Accelerating Graph Pattern Matching using PIM with Optimized Cache System

Liang Yan[1,2], Xiaoyang Lu[3], Xiaoming Chen[1,2], Sheng Xu[4], Xingqi Zou[1,2], Yinhe Han[1,2], Xian-He Sun[3]

[1] Institute of Computing Technology, Chinese Academy of Sciences

[2] University of Chinese Academy of Sciences

[3] Illinois Institute of Technology

[4] Anhui Normal University

Outline

- Background
  - Graph Pattern Matching (GPM)
  - Processing-in-memory (PIM)
- Challenges
- AceMiner Design
- Results

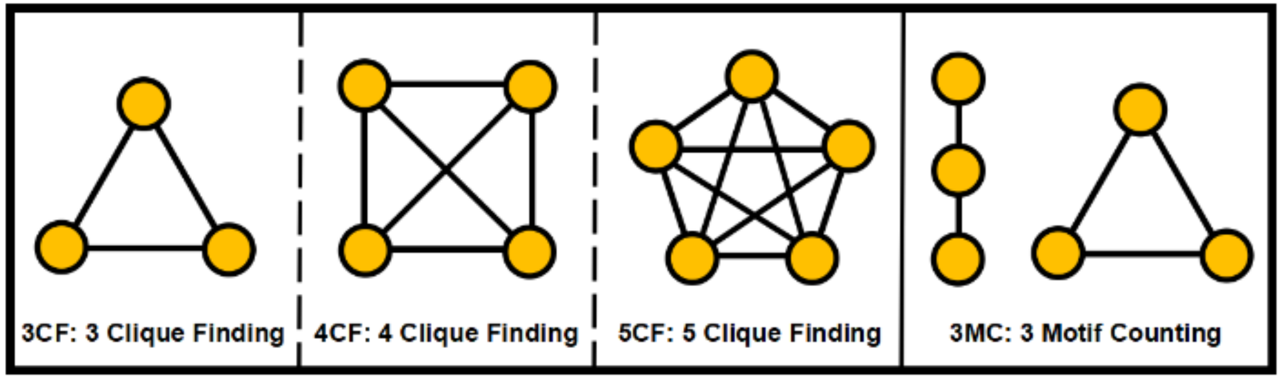
Background

☐ Big Graphs

- 2 Billion Facebook users
- 3 Billion base pairs in human genome
- 20 Billion internet connected devices
- Trillions of connections between them

☐ Graph Pattern Matching

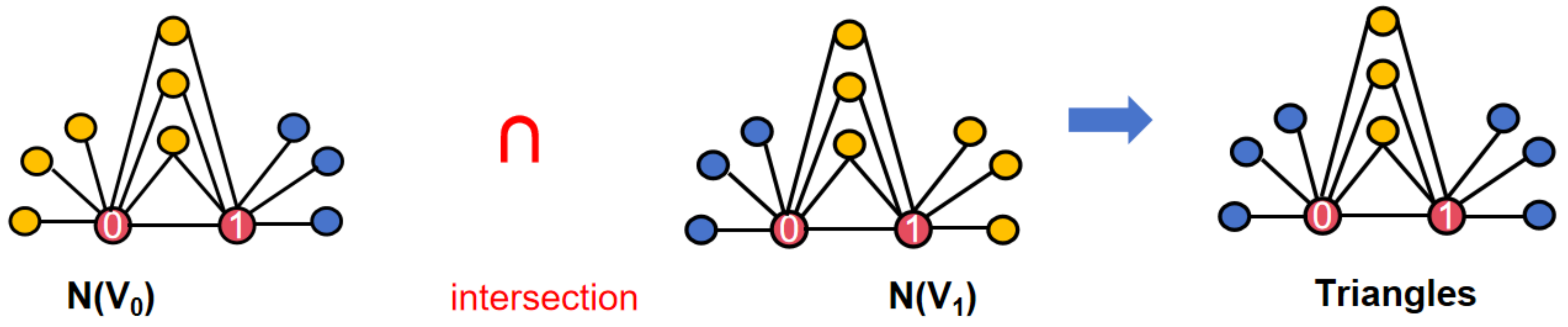
- Aims to discover *structural patterns* in a graph



Background

☐ Graph Pattern Matching

- Triangle Counting：

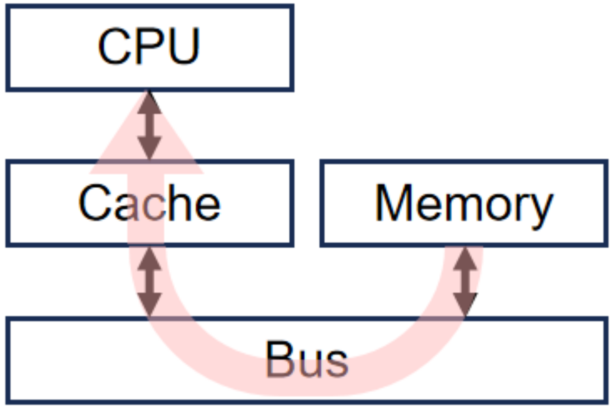


Graph pattern matching algorithms involve a lot of:

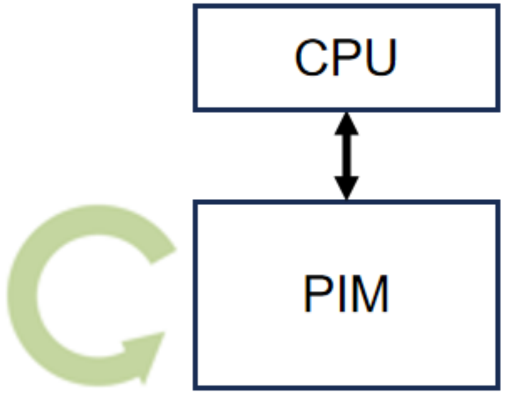
- access to neighbouring vertex set
- irregular memory accesses

Background

☐ Processing in memory



von Neumann architecture



PIM architecture

weak performance!

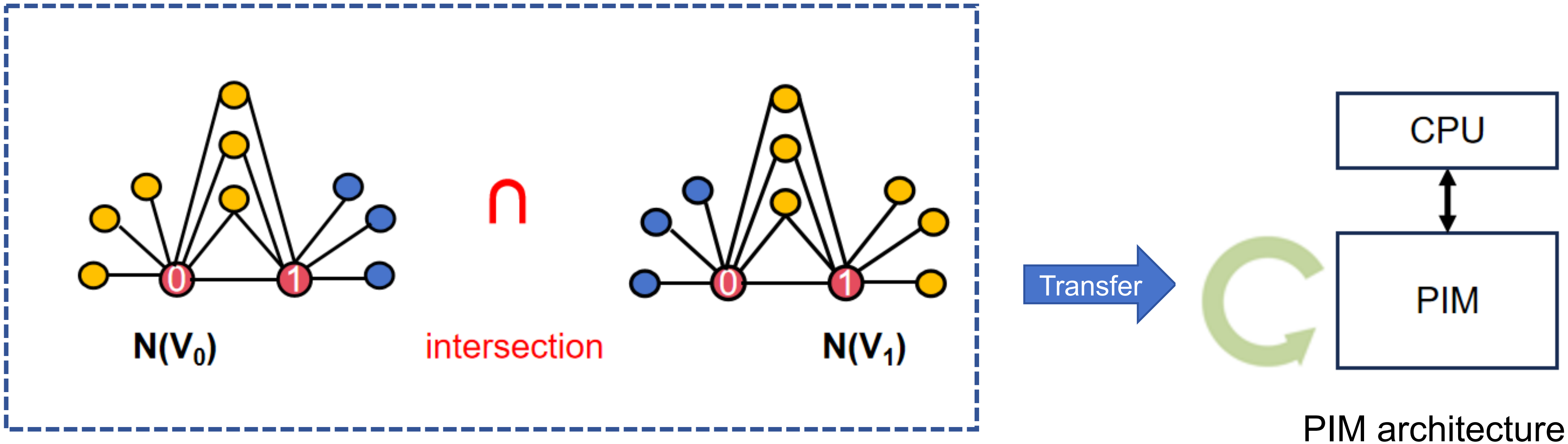
Data moves from memory to processor → near data processing

Data moves from memory to processor		near data processing
High memory access energy	→	Low memory access energy
Limited transmission bandwidth	→	High transmission bandwidth
Large data transmission	→	Minor data transmission

Background

GPM + PIM

- The set operation does not need too much computing resource
- Transfer all the set operations of graph pattern matching into PIM



PIM architecture

Outline

- Background
  - Graph Mining
  - Processing-in-memory (PIM)
- Challenges
- AceMiner Design
- Results & Conclusions

☐ GPM + PIM： Challenges

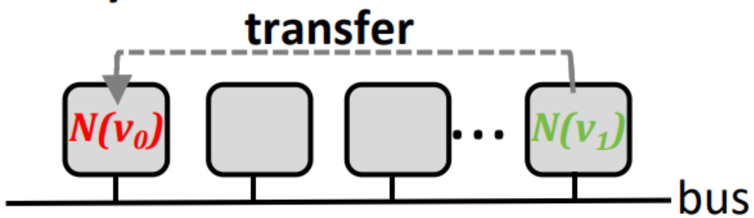
Triangle Counting:

```
procedure GPM_TC(G,P)
  for v0 ∈ V
    for v1 ∈ N(v0)① and v1 > v0③
      for v2 ∈ N(v0) ∩ N(v1)② and v2 > v1
        (v0, v1, v2) is an subgraph
          subgraph += 1
```

①：**Potential locality**

$N(v_0)$ is **reused** in two continous iterations

②：**Heavy data movement**

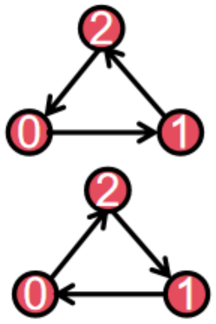


③：**Heavy comparison overhead**

if $v_0 = 10$, $v_1 > v_0$

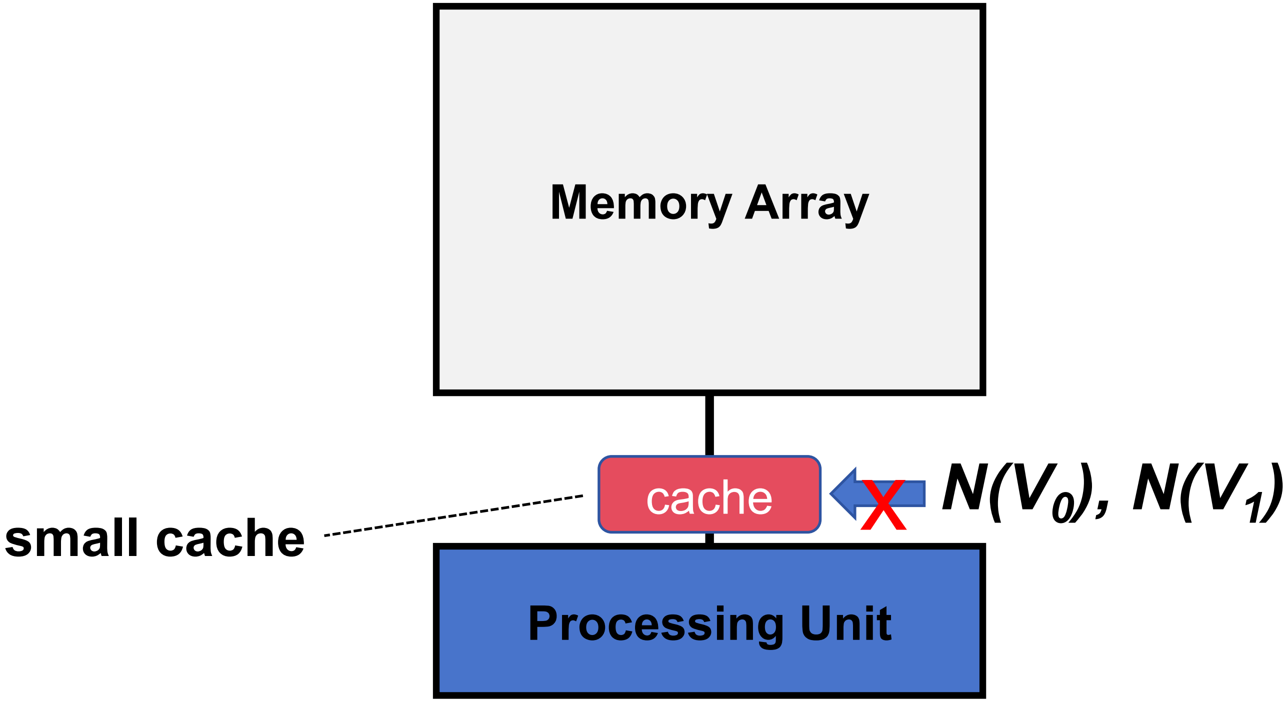
$v_1 \in N(v_0) = \{0, 1, 3, 4, 6, 7, 8, 9, 11\}$

discarded



All these problems can be mitigated by introducing a new cache system in the PIM architecture!!

GPM + PIM: Challenges




```
procedure GPM_TC(G,P)
  for v0 ∈ V
    for v1 ∈ N(v0) ① and v1 > v0 ③
      for v2 ∈ N(v0) ∩ N(v1) ② and v2 > v1
        (v0, v1, v2) is an subgraph
          subgraph += 1
```

Existing PIM can support a very limited size of the cache :

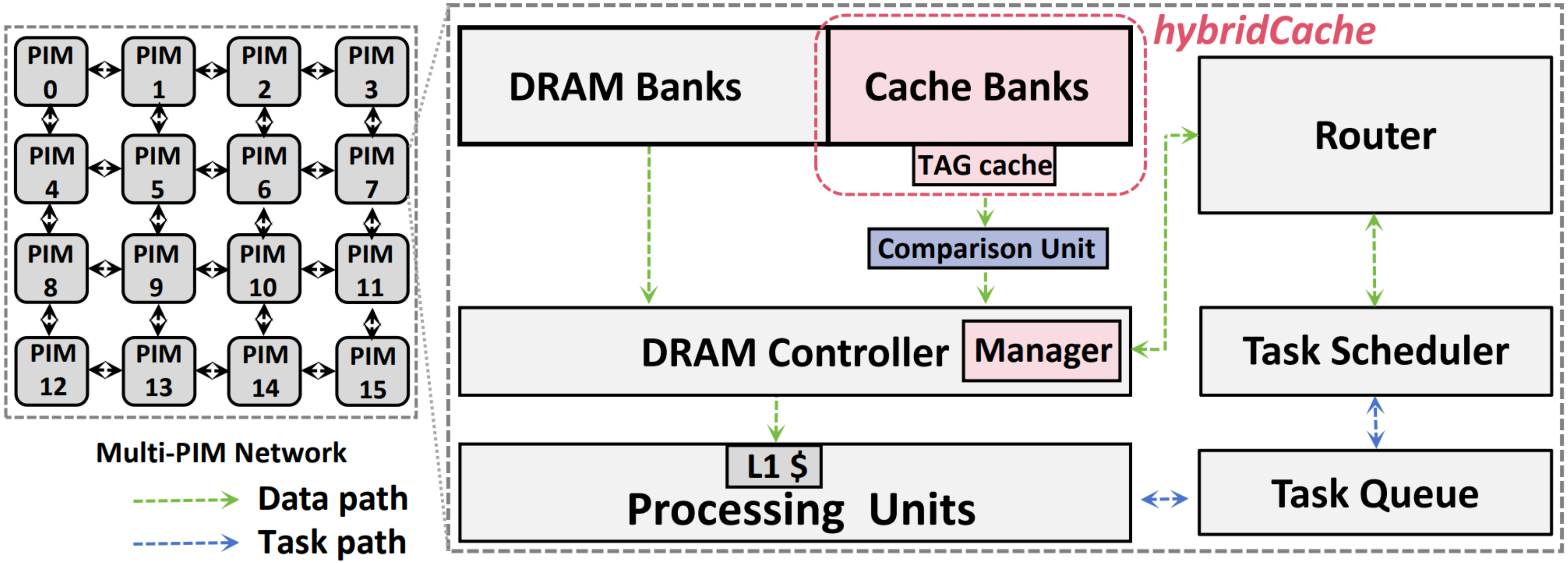
- But the size of the set of neighbouring vertice is random
- N(v0) of successive accesses are likely to be excluded from the cache
- N(v1) from remote devices cannot be cached locally

Outline

- Background
  - Graph Mining
  - Processing-in-memory (PIM)
- Challenges
- AceMiner Design
- Results & Conclusions

AceMiner Design

Overall Architecture :



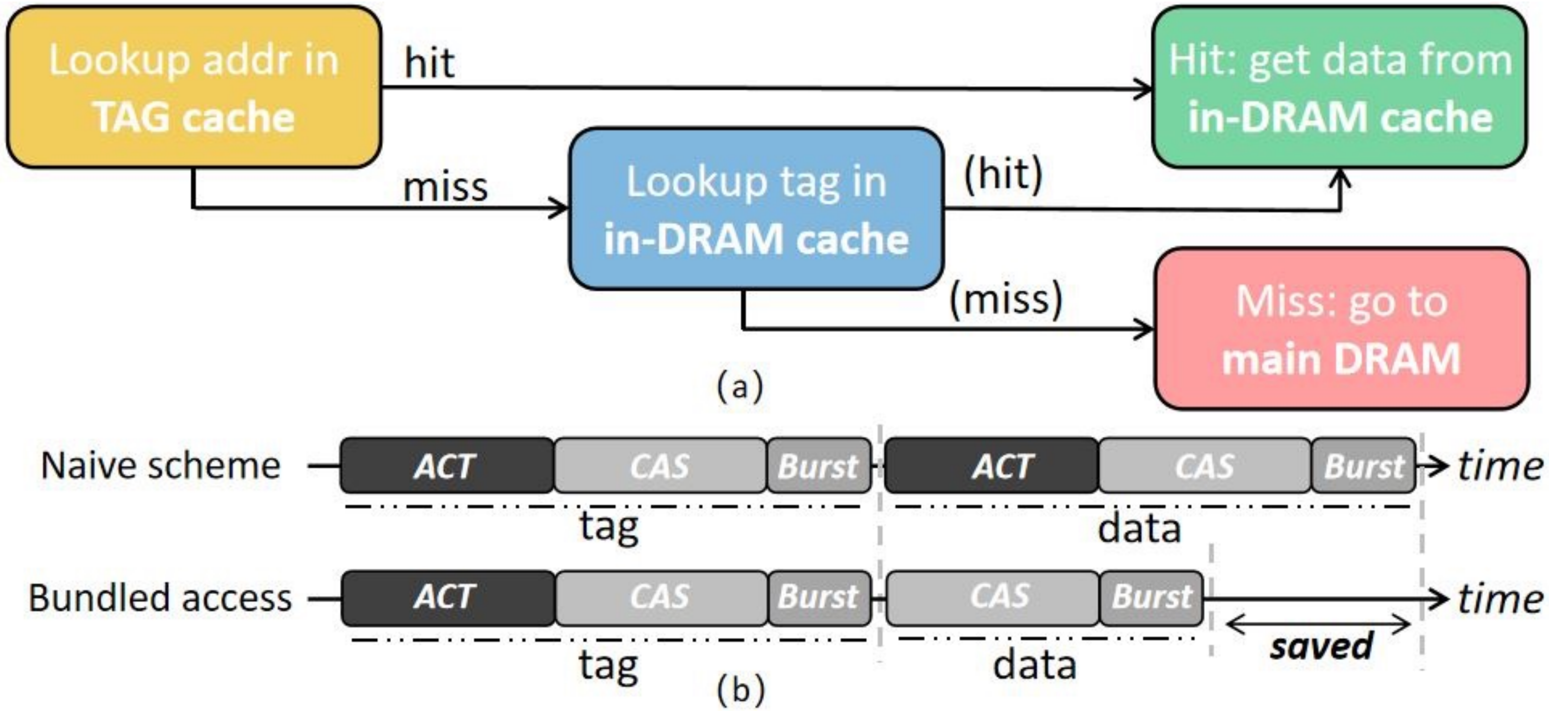
Introducing in-DRAM cache to meet the large-capacity demand of GPM :

- Introducing **TAG cache** to accelerate accesses to hybridCache system
- Introducing an **optimised cache replacement strategy** to improve efficiency
- Seamless integration of **comparison unit** design with hybridCache

AceMiner Design

☐ Access Latency Optimization:

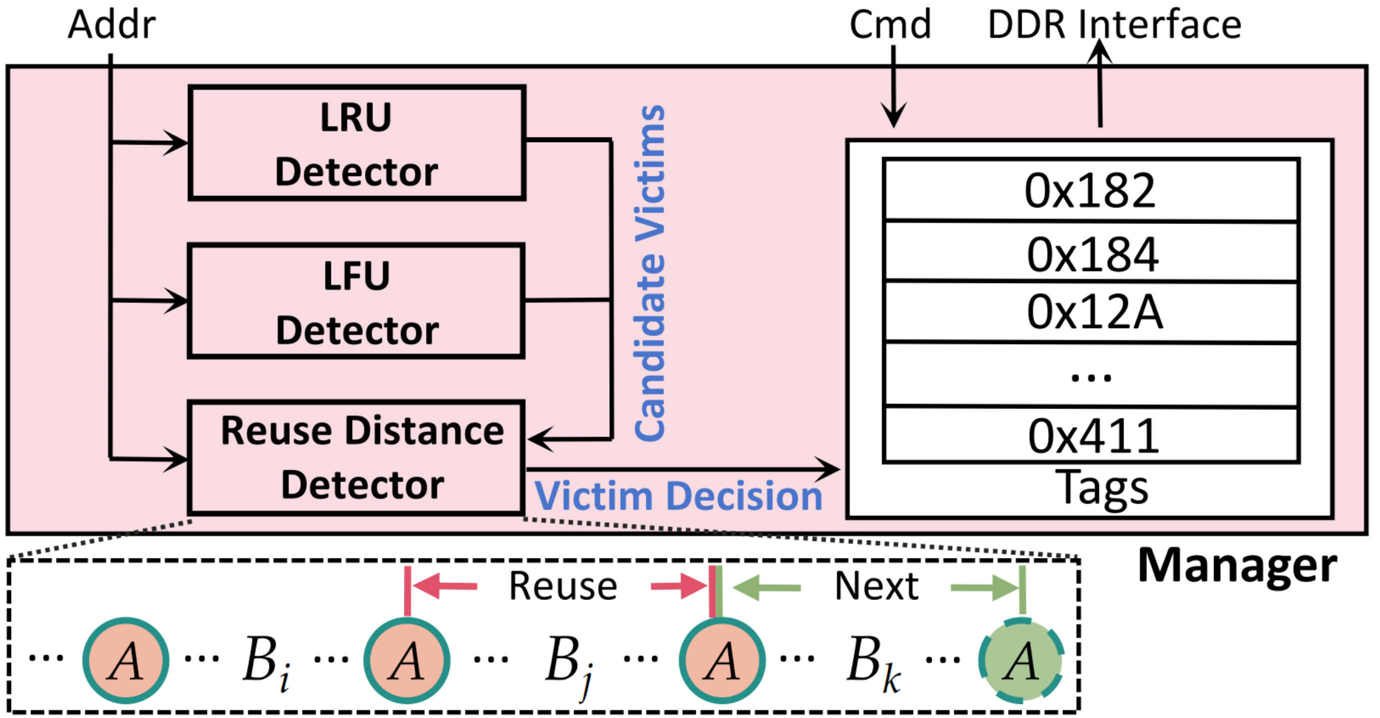
- Introducing SRAM-based TAG cache to cache frequently accessed in-DRAM cache tags
- Adopt bundled access method to package in-DRAM cache tag and data accesses



AceMiner Design

◻ Advanced Replacement Policy：

- Existing replacement strategies Least Recently Used (LRU) or Least Frequently Used (LFU) are deficient
- For graph pattern matching applications with extremely irregular access characteristics, these recency-based approaches may ignore some **recent infrequent but globally frequent data**

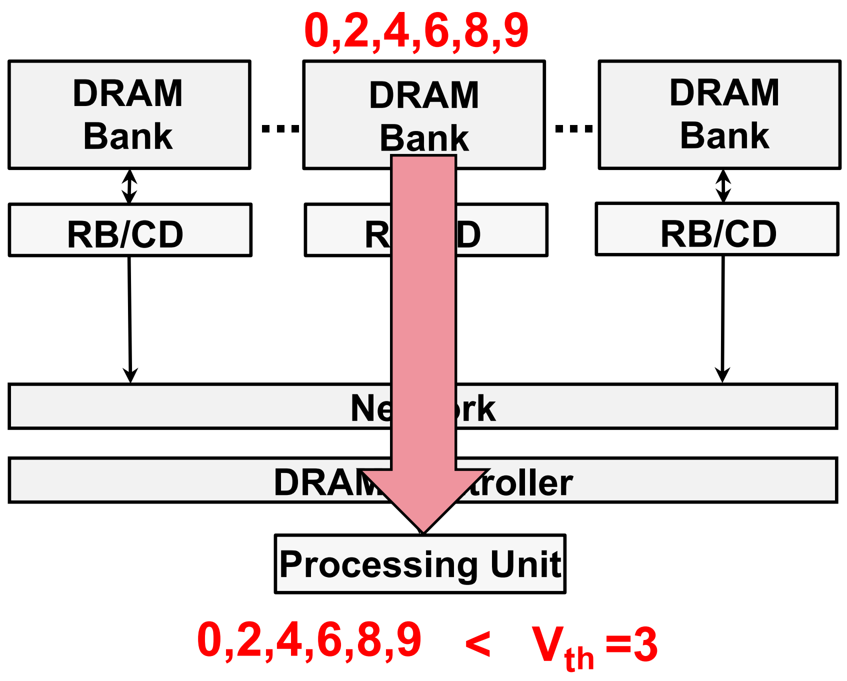


Reuse distance based replacement policy

AceMiner Design

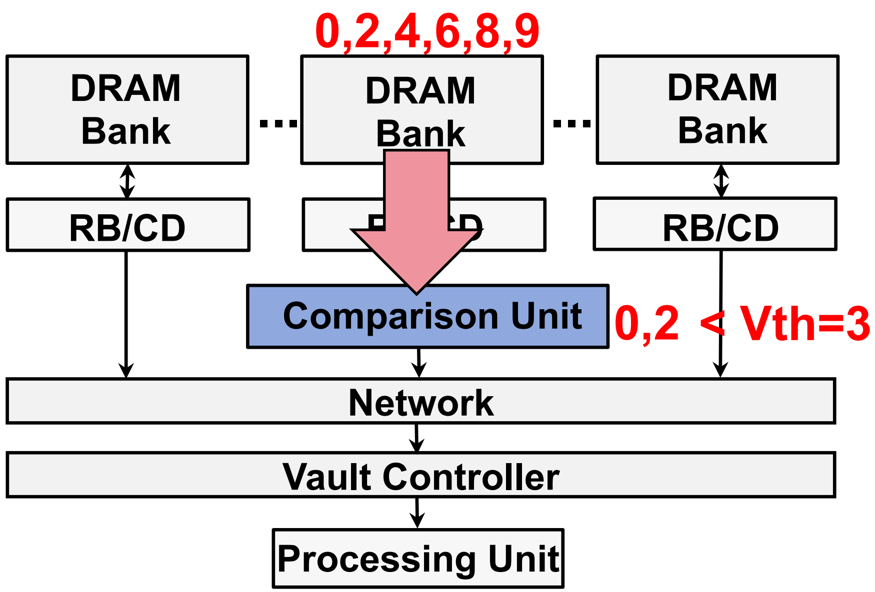
☐ Seamless Integration of Comparison Units

‘Compare Parts’ in processing unit



The data needs to be fully loaded to the PU to do comparison

‘Compare parts’ after row buffer.

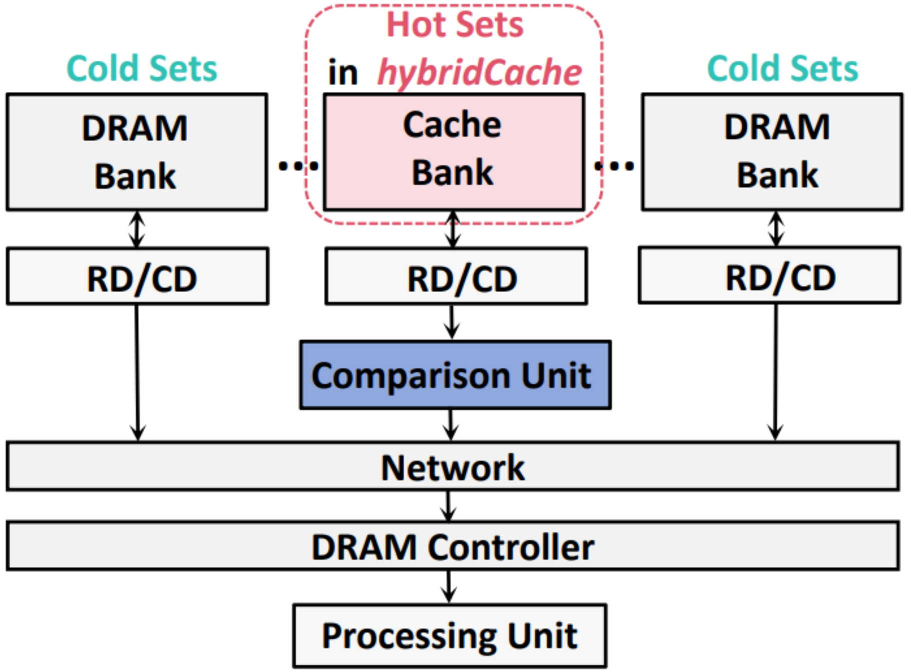


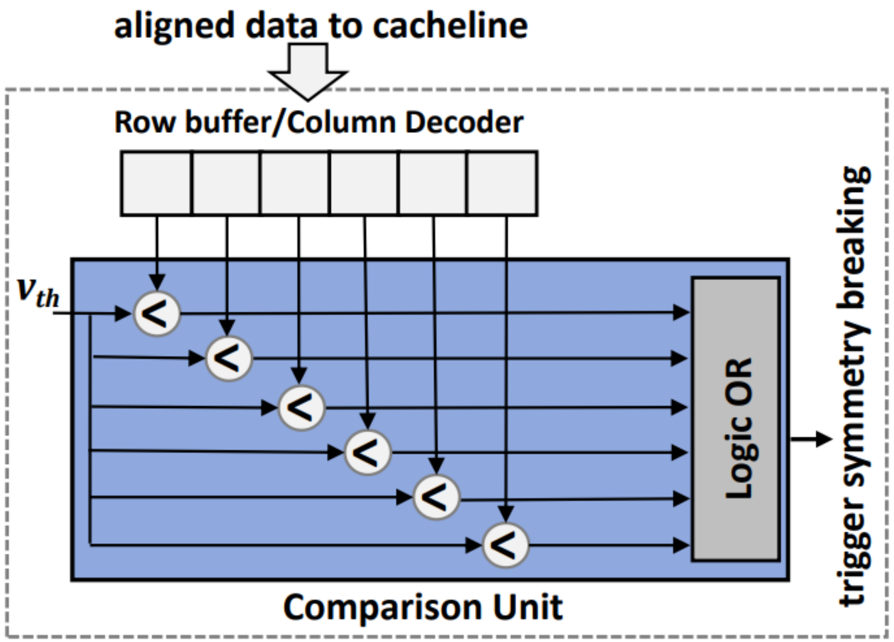
More timely comparison operations

AceMiner Design

☐ Seamless Integration of Comparison Units

- In triangle finding, the top 10% frequently visited neighbouring vertex sets account for nearly 44% of the total visits. Configuring the comparison unit at each bank increases the area overhead





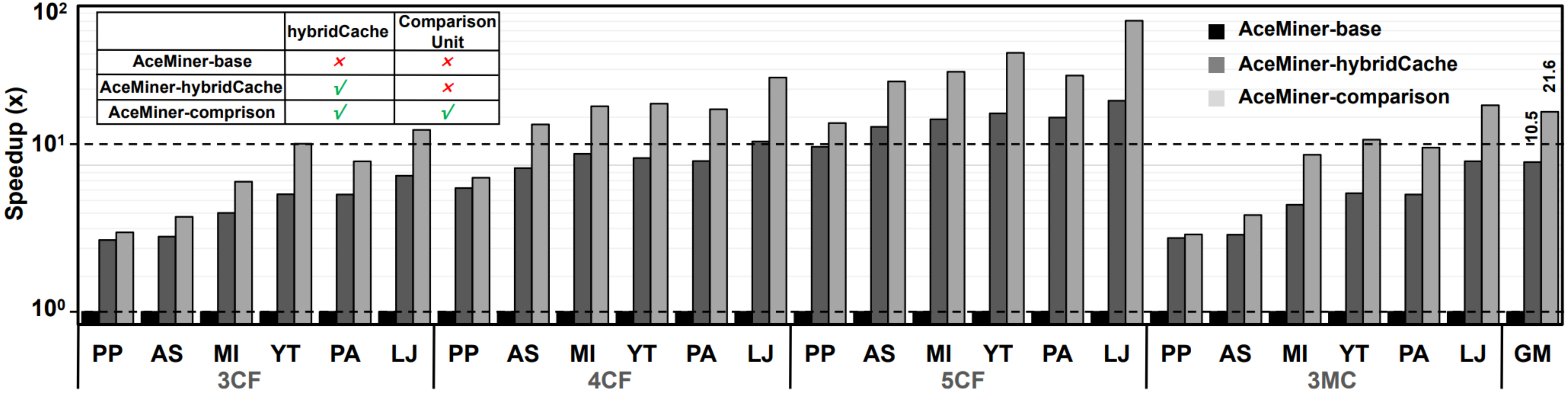
Outline

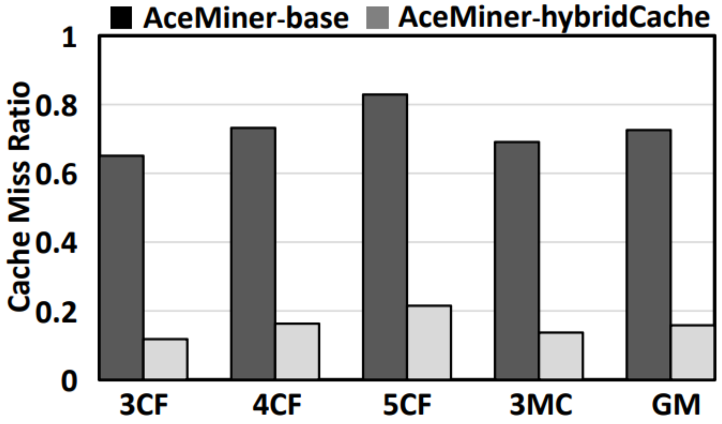
- Background
  - Graph Mining
  - Processing-in-memory (PIM)
- Challenges
- AceMiner Design
- Results

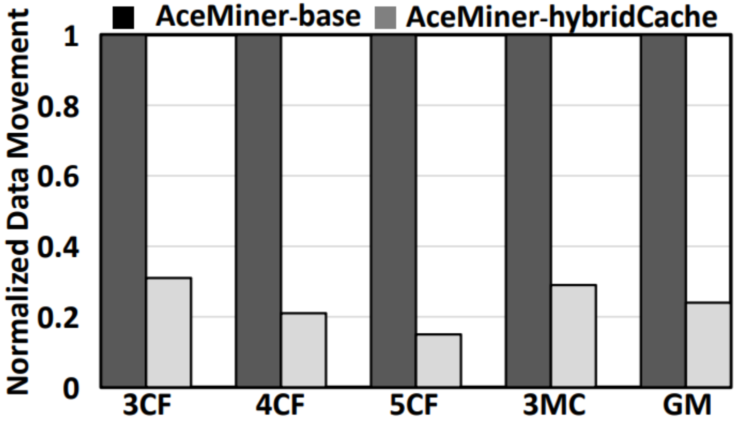
Results

☐ Results:

- AceMiner 21.6x performance improvement







Performance gains are mainly due to improved data access locality and reduced data movement.

Results

Results:

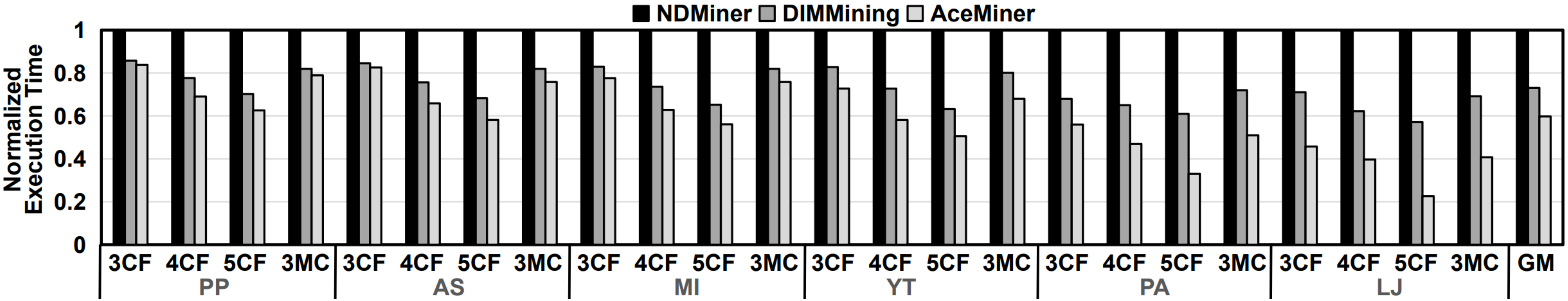
- AceMiner outperforms the state-of-the-art, achieving speedups of 40.2% and 13.3% over NDMiner and DIMMining respectively, with less energy consumption and design overhead

GPM ACCELERATING WITH PIM FRAMEWORKS.

	Comparison OPT	PIM logic	PIM cache
NDMiner	hardware	dedicated	small cache
DIMMining	software	dedicated	small cache
AceMiner	hardware	general	*hybridCache*

DESIGN OVERHEAD COMPARISON.

	NDMiner	DIMMining	AceMiner
Area	0.64 mm^2	0.38 mm^2	0.11 mm^2
Power	51.59 mW	105.82 mW	10.85 mW



Thanks for your attentions !

Q&A