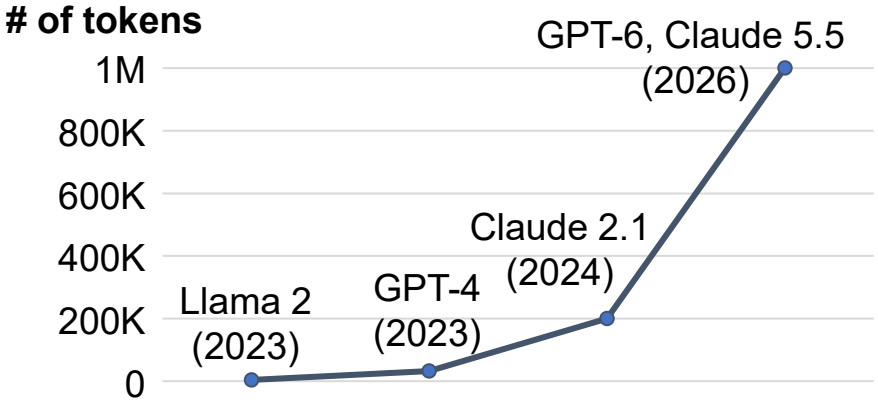
A Hybrid Processing-in-Memory Architecture for Long Sequence LLM Inference with KV Cache Filtering

Soojin Hwang[1], Sanghyeon Lee[2],
Juhyun Lee[2], Jaehyuk Huh[2]

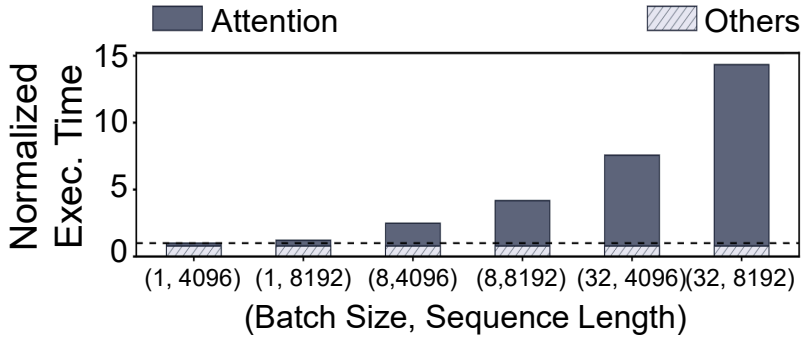
ETRI[1], KAIST[2]

Long Sequence LLM Inference

- Sequence (context) length keeps growing
- Attention computation time increases



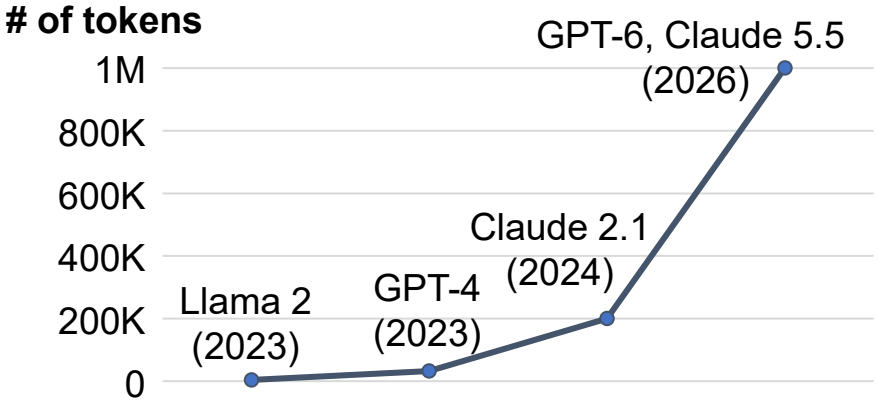
Maximum context window of modern LLMs



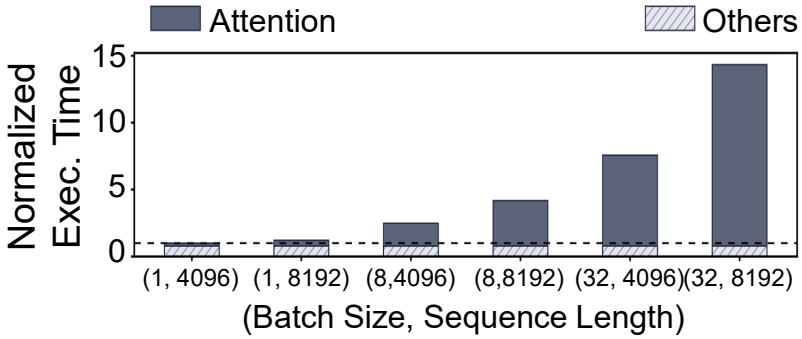
Portion of attention computation
In NPU-PIM system

Long Sequence LLM Inference

- Sequence (context) length keeps growing
- Attention computation time increases



Maximum context window of modern LLMs



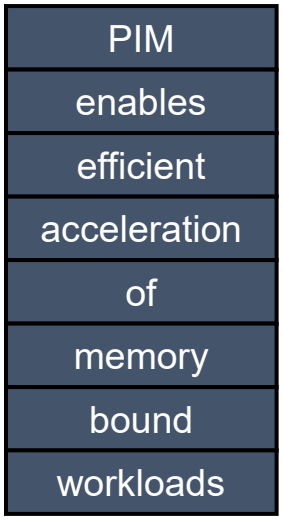
Portion of attention computation
In NPU-PIM system

Need to reduce the amount of attention computation:
Sparse attention is a promising approach

Dynamic KV Cache Filtering

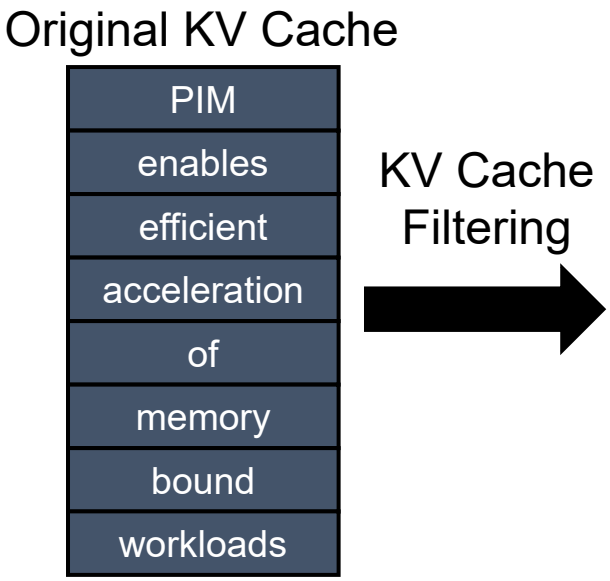
- Decoding with KV cache entry of **critical tokens**
- Enables **sparse attention** with dense data
- Highly **memory-intensive, irregular** kernels

Original KV Cache



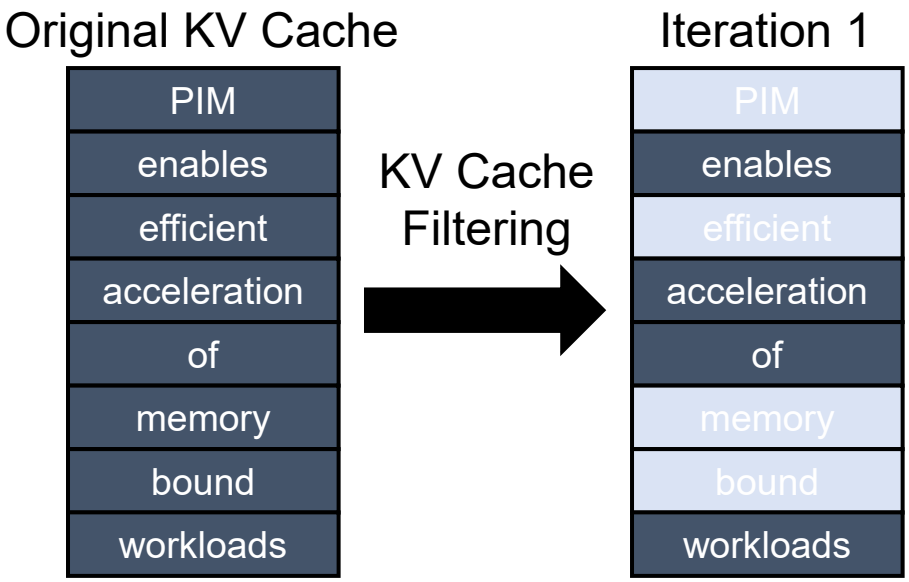
Dynamic KV Cache Filtering

- Decoding with KV cache entry of **critical tokens**
- Enables **sparse attention** with dense data
- Highly **memory-intensive, irregular** kernels



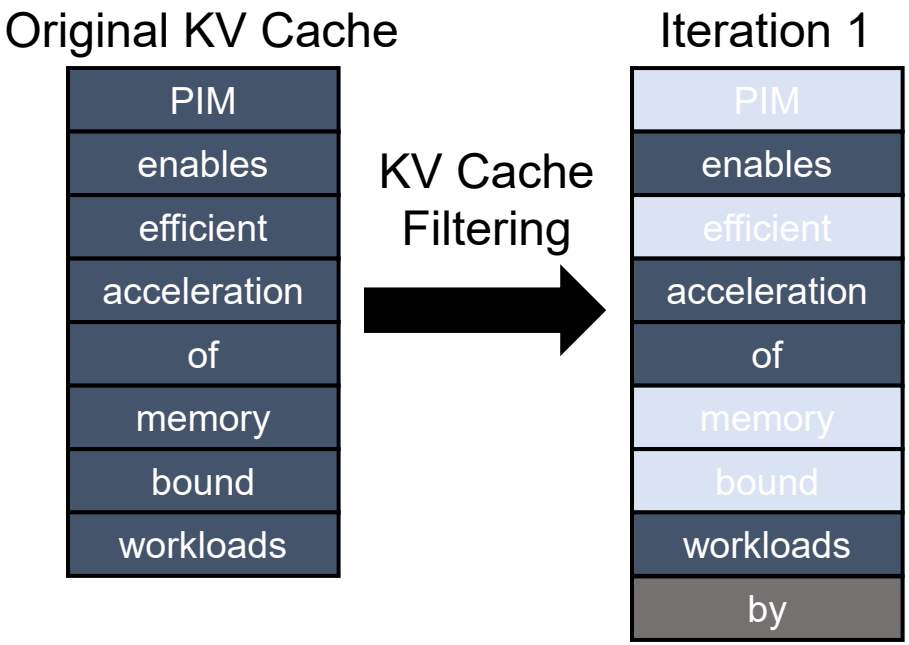
Dynamic KV Cache Filtering

- Decoding with KV cache entry of **critical tokens**
- Enables **sparse attention** with dense data
- Highly **memory-intensive, irregular** kernels



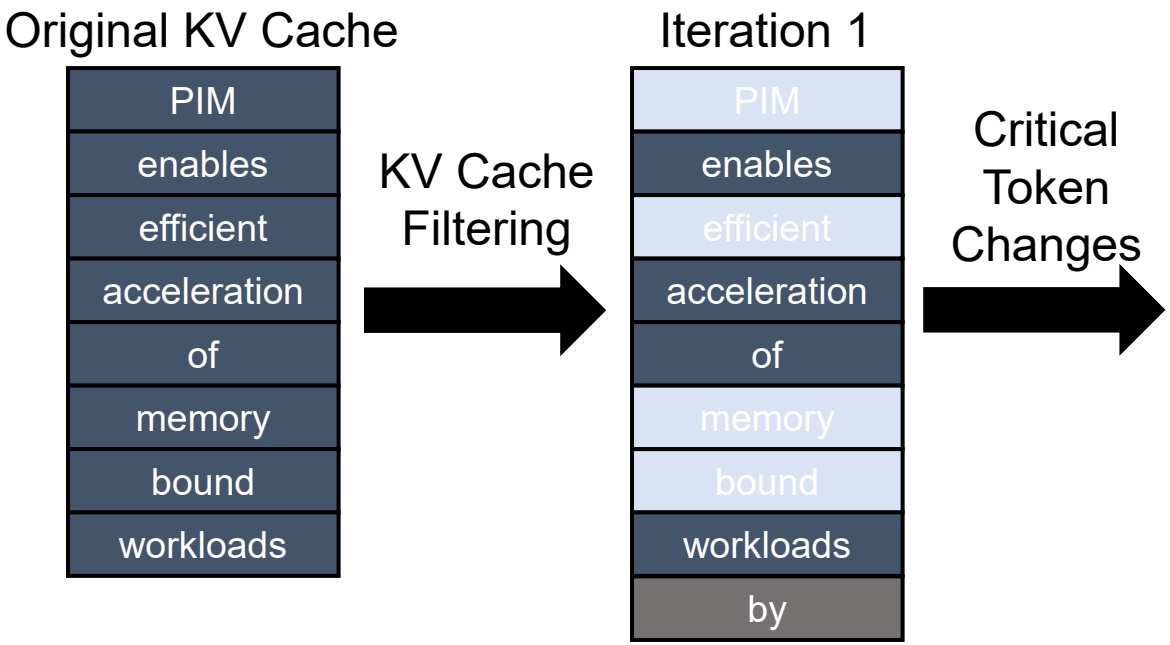
Dynamic KV Cache Filtering

- Decoding with KV cache entry of **critical tokens**
- Enables **sparse attention** with dense data
- Highly **memory-intensive, irregular** kernels



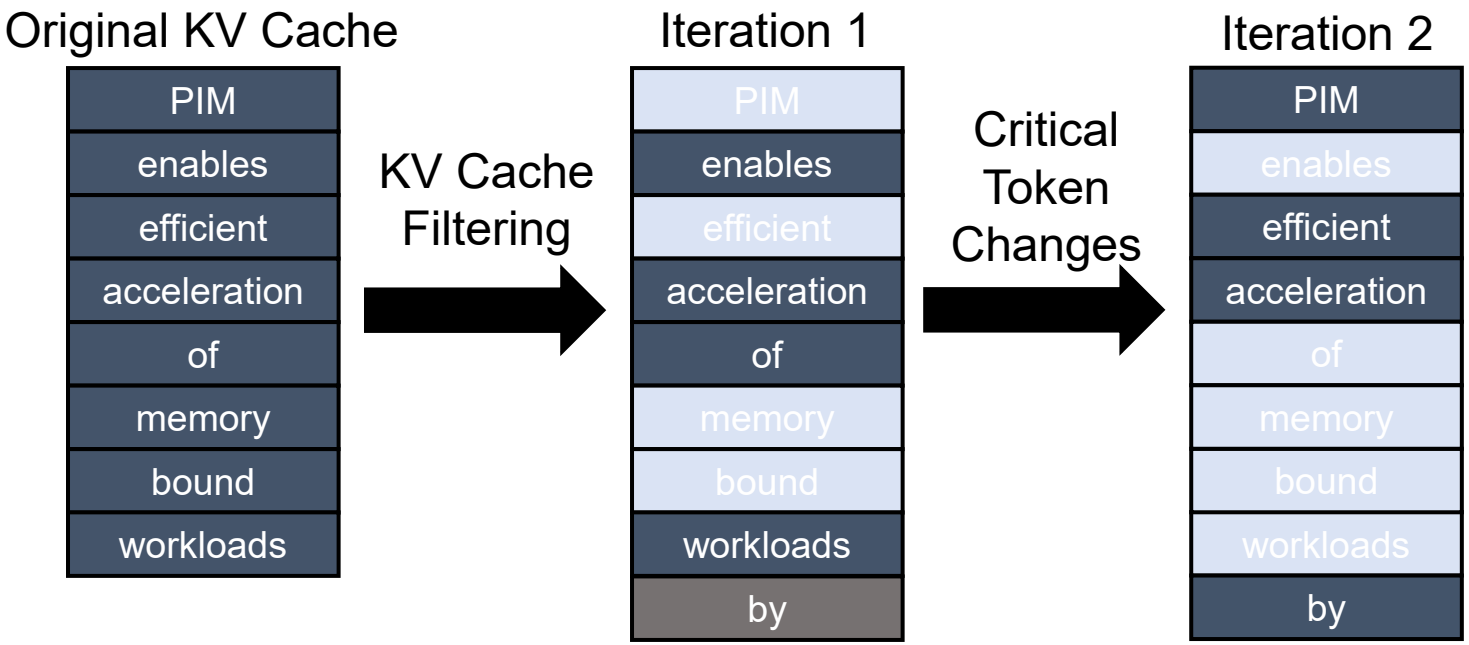
Dynamic KV Cache Filtering

- Decoding with KV cache entry of **critical tokens**
- Enables **sparse attention** with dense data
- Highly **memory-intensive, irregular** kernels



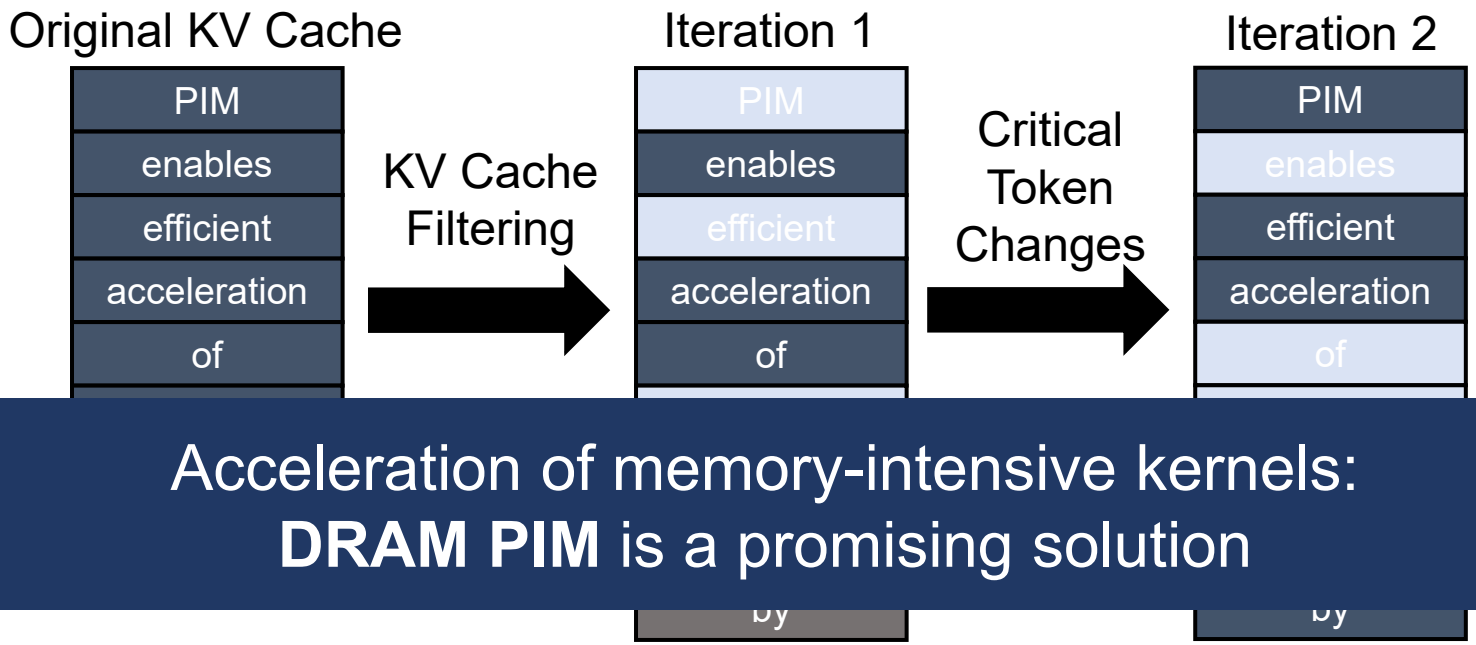
Dynamic KV Cache Filtering

- Decoding with KV cache entry of **critical tokens**
- Enables **sparse attention** with dense data
- Highly **memory-intensive, irregular** kernels



Dynamic KV Cache Filtering

- Decoding with KV cache entry of **critical tokens**
- Enables **sparse attention** with dense data
- Highly **memory-intensive, irregular** kernels

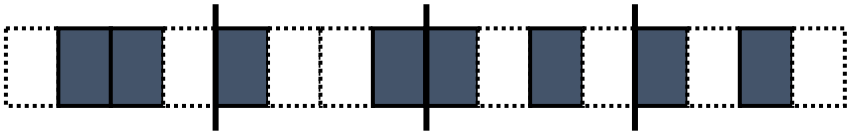


Acceleration of memory-intensive kernels:
DRAM PIM is a promising solution

Research Goal

Design a **DRAM PIM** Architecture for
Memory-intensive and Irregular **Sparse Attention**

Sparse Attention
(KV Cache)

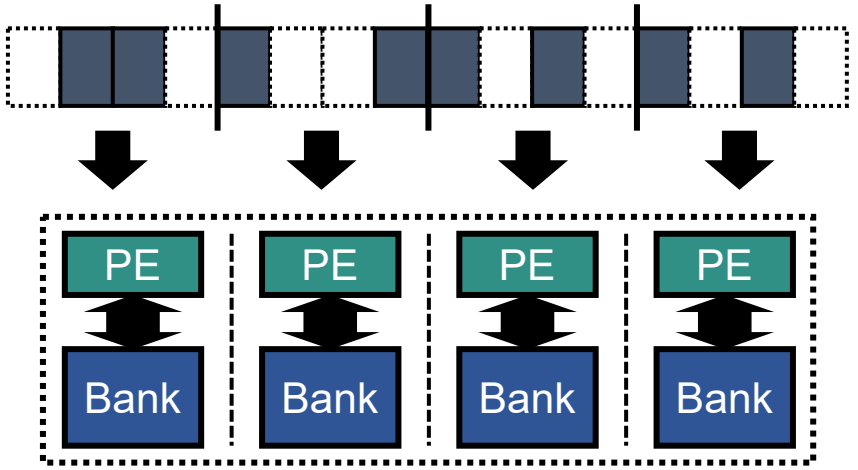


Research Goal

Design a **DRAM PIM** Architecture for
Memory-intensive and Irregular **Sparse Attention**

Sparse Attention
(KV Cache)

Dynamic Sparsity-aware
DRAM PIM Design



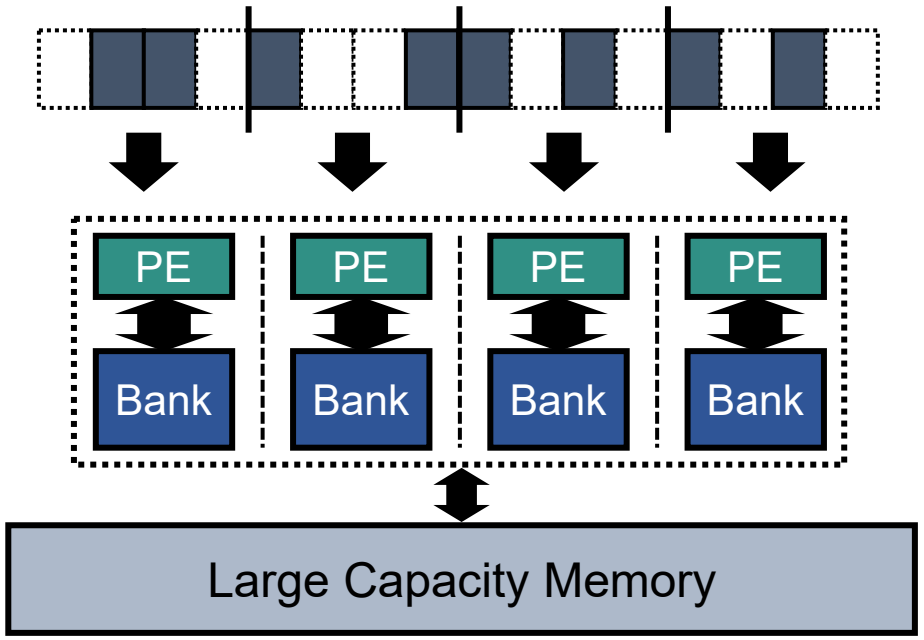
Research Goal

Design a **DRAM PIM** Architecture for
Memory-intensive and Irregular **Sparse Attention**

Sparse Attention
(KV Cache)

Dynamic Sparsity-aware
DRAM PIM Design

Extension to Hybrid Memory
for storing whole KV cache



Contents

- Introduction
- All-bank DRAM PIM
- Challenges
- Design: PASCAL
- Evaluation & Conclusion

LLM Inference with xPU-PIM System

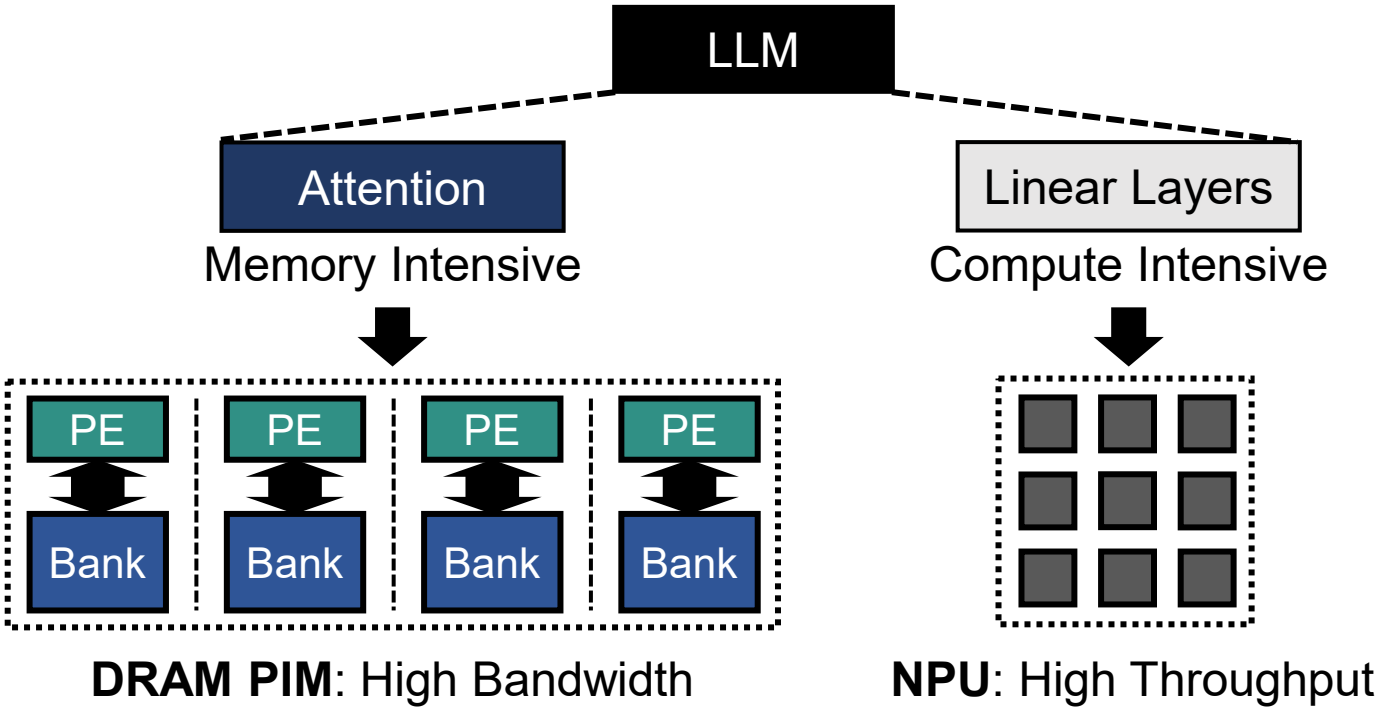
- Map kernels by memory intensiveness[1, 2]



[1] Heo et al., “NeuPIMs: A NPU-PIM Heterogeneous Acceleration for Batched Inference of Large Language Model”, ASPLOS 2024.
[2] Park et al., “AttAcc!: Unleashing the Power of PIM for Batched Transformer-based Generative Model Inference”, ASPLOS 2024.

LLM Inference with xPU-PIM System

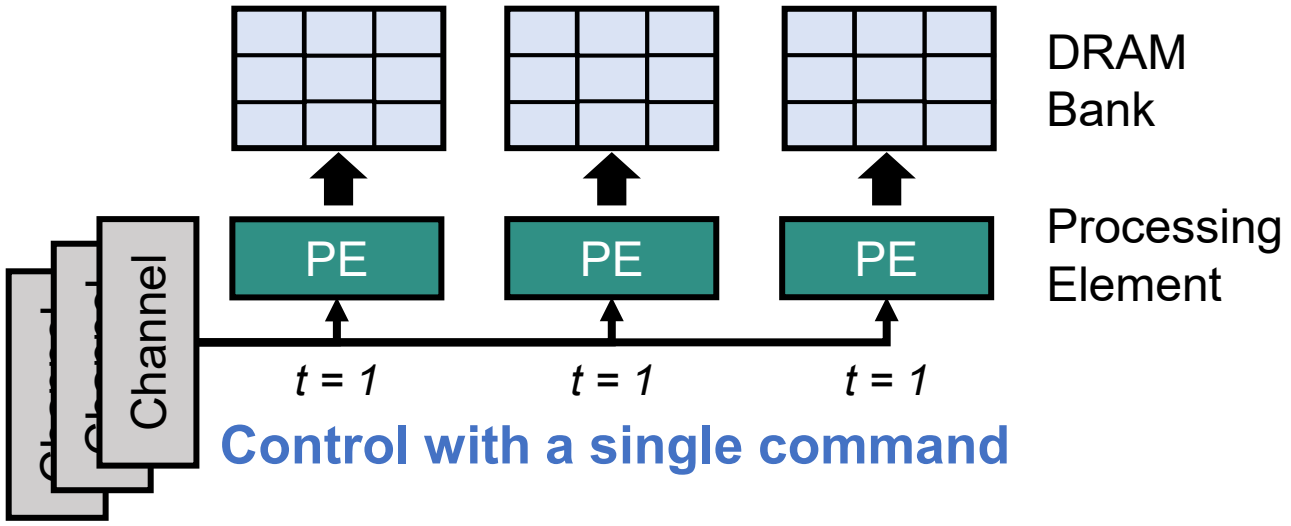
- Map kernels by memory intensiveness[1, 2]
- All-bank DRAM PIM for attention acceleration



[1] Heo et al., “NeuPIMs: A NPU-PIM Heterogeneous Acceleration for Batched Inference of Large Language Model”, ASPLOS 2024.
[2] Park et al., “AttAcc!: Unleashing the Power of PIM for Batched Transformer-based Generative Model Inference”, ASPLOS 2024.

All-bank DRAM PIM[1,2]

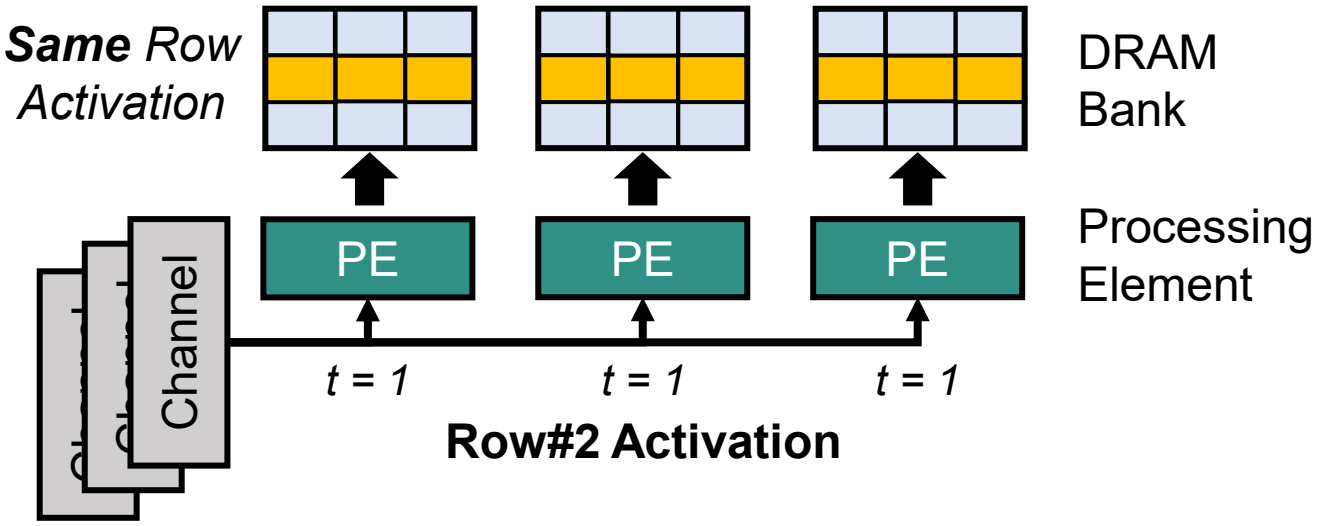
- Direct data transfer from bank to PE
- **All banks** under a channel **share commands**
- Efficiently exploits **DRAM internal bandwidth**
  - Through bank-level parallelism



[1] He et al., “Newton: A DRAM-maker’s Accelerator-in-Memory (AiM) Architecture for Machine Learning”, MICRO 2020.
[2] Lee et al., “Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology”, ISCA 2021.

All-bank DRAM PIM[1,2]

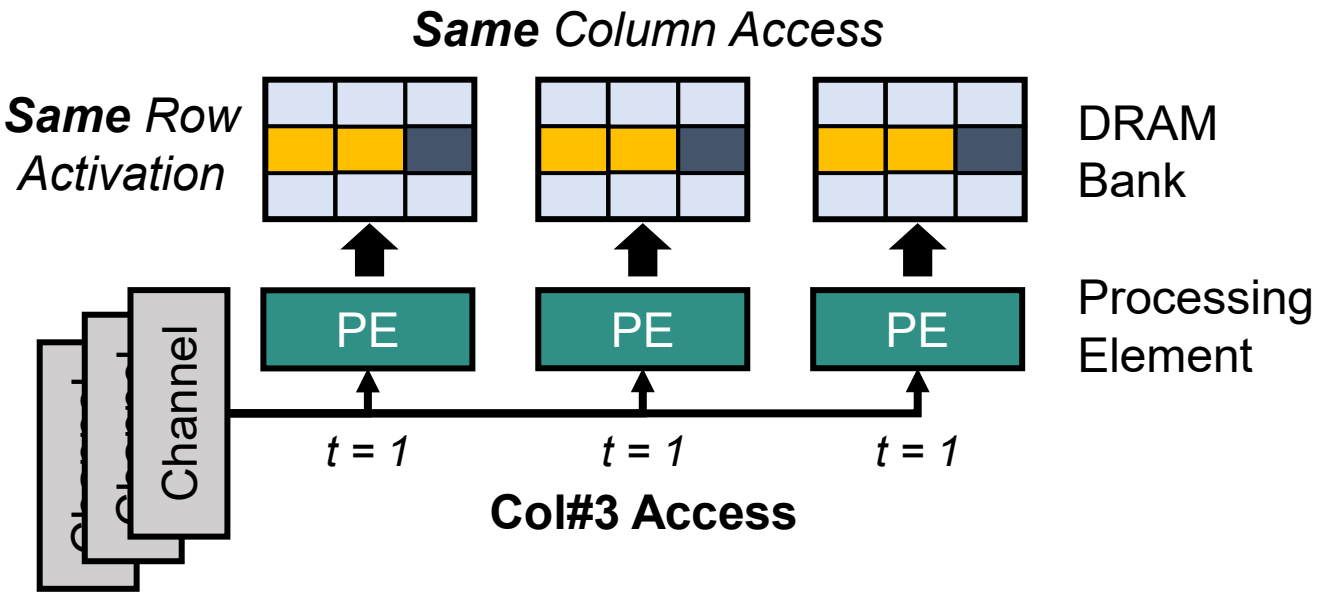
- Direct data transfer from bank to PE
- **All banks** under a channel **share commands**
- Efficiently exploits **DRAM internal bandwidth**
  - Through bank-level parallelism



[1] He et al., "Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning", MICRO 2020.
[2] Lee et al., "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology", ISCA 2021.

All-bank DRAM PIM[1,2]

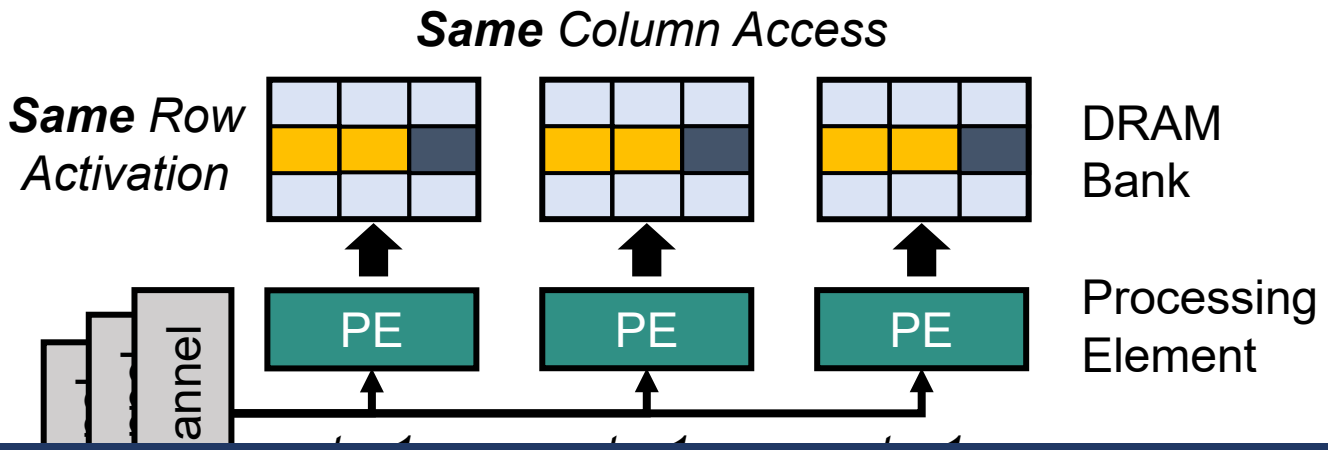
- Direct data transfer from bank to PE
- **All banks** under a channel **share commands**
- Efficiently exploits **DRAM internal bandwidth**
  - Through bank-level parallelism



[1] He et al., "Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning", MICRO 2020.
[2] Lee et al., "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology", ISCA 2021.

All-bank DRAM PIM[1,2]

- Direct data transfer from bank to PE
- **All banks** under a channel **share commands**
- Efficiently exploits **DRAM internal bandwidth**
  - Through bank-level parallelism



Selective operand access is not allowed

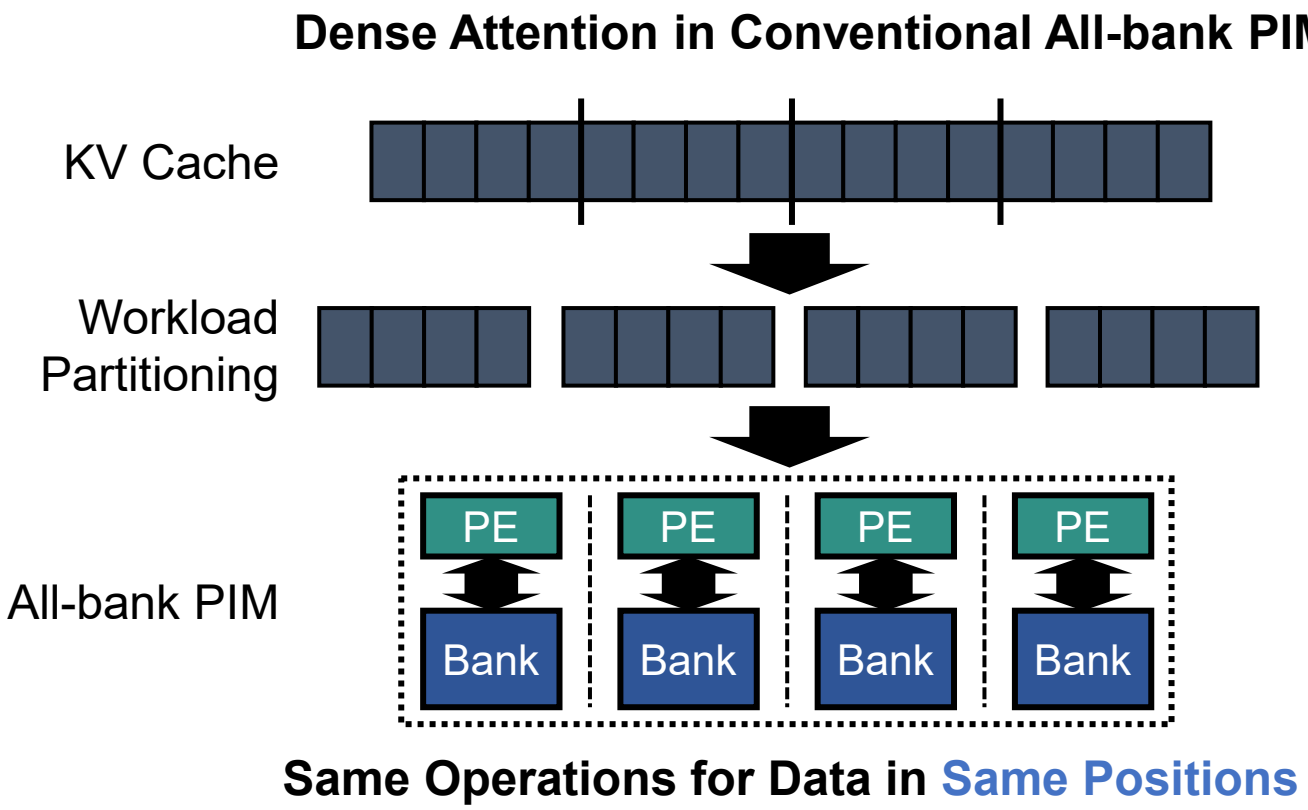
[1] He et al., "Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning", MICRO 2020.
[2] Lee et al., "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology", ISCA 2021.

Contents

- Introduction
- All-bank DRAM PIM
- Challenges
- Design: PASCAL
- Evaluation & Conclusion

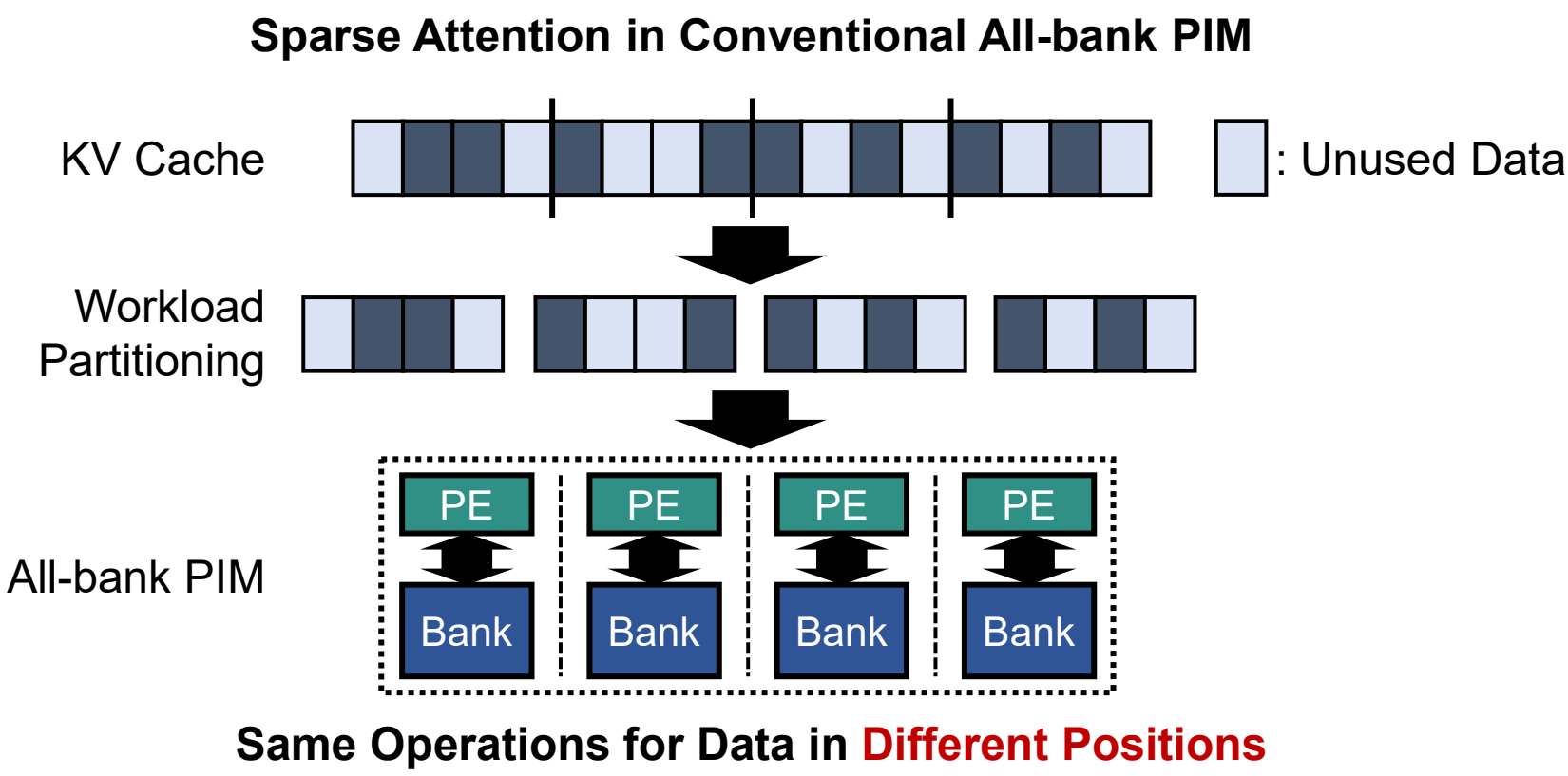
All-bank PIM with KV Cache Filtering

- Challenge: Sparsity & Dynamism



All-bank PIM with KV Cache Filtering

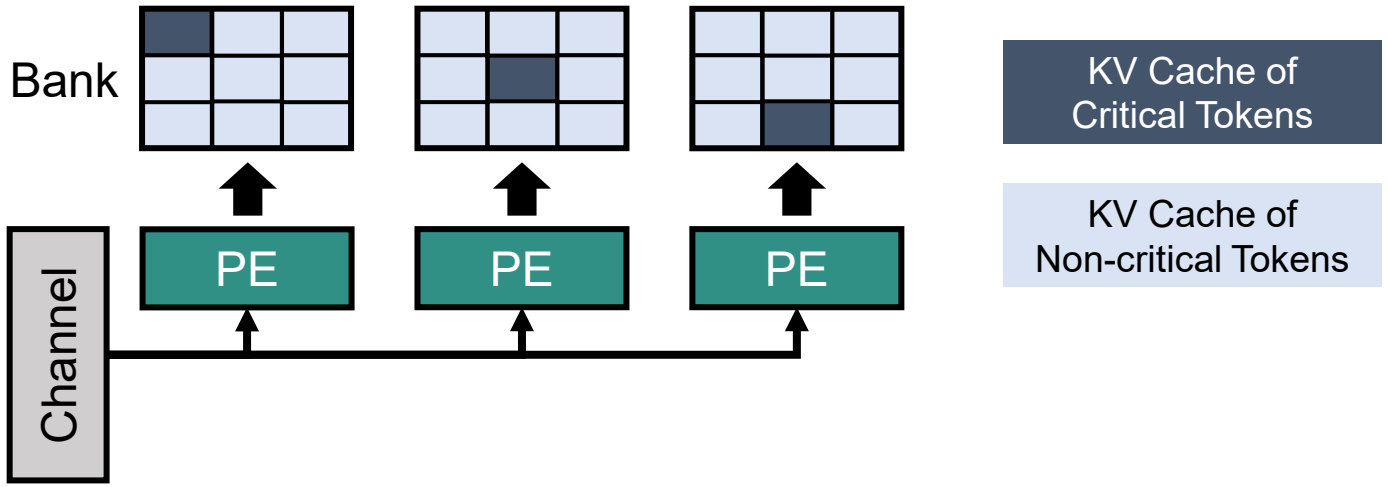
- Challenge: Sparsity & Dynamism



Challenge 1. Data Divergence

- Incurred by different position of operands
- Need access to different row and column addresses
- Requires distinct operation in each bank

Requirement:
Read **critical token** (dark cell) in each bank

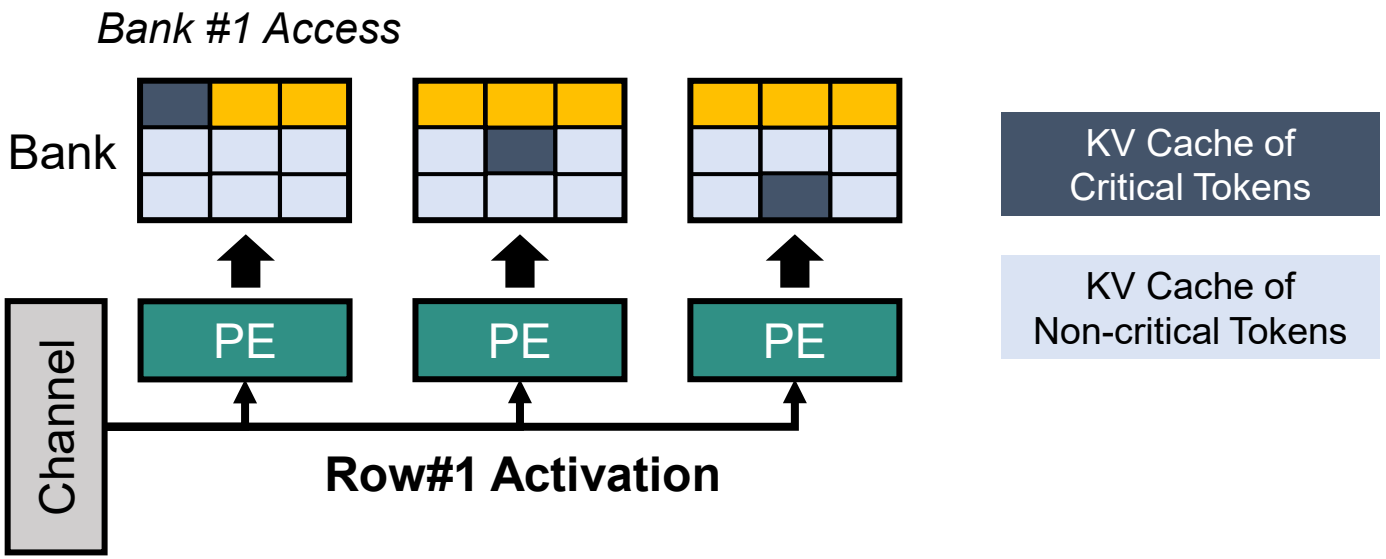


Challenge 1. Data Divergence

- Incurred by different position of operands
- Need access to different row and column addresses
- Requires distinct operation in each bank

All-bank PIM:

Redundant multiple accesses (row activations)

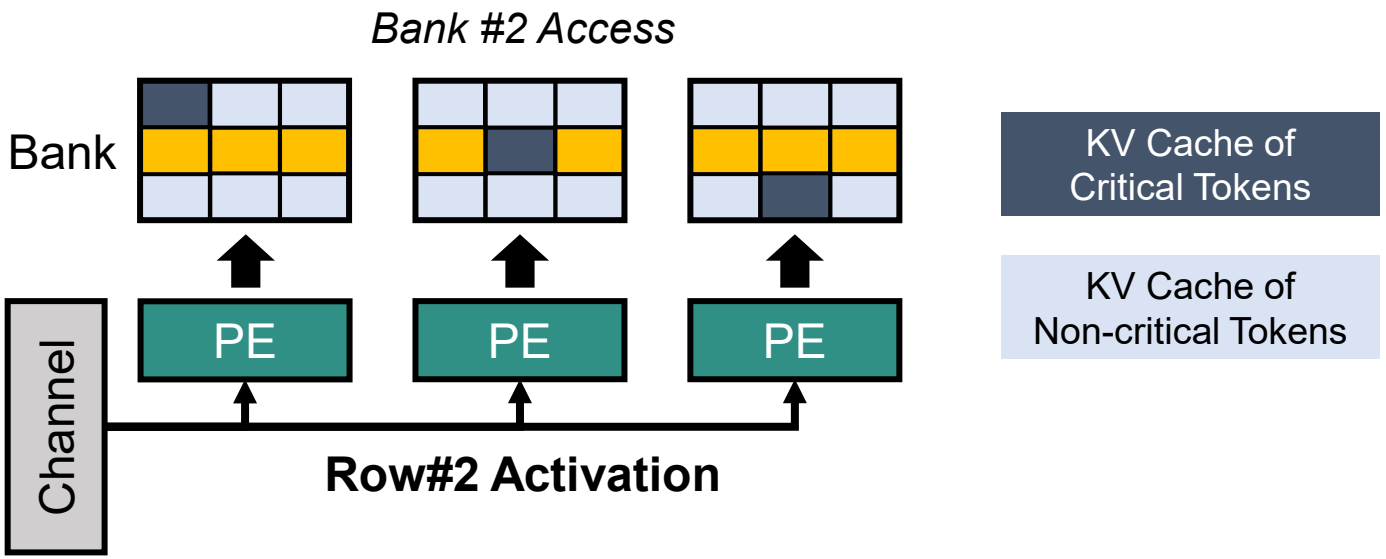


Challenge 1. Data Divergence

- Incurred by different position of operands
- Need access to different row and column addresses
- Requires distinct operation in each bank

All-bank PIM:

Redundant multiple accesses (row activations)

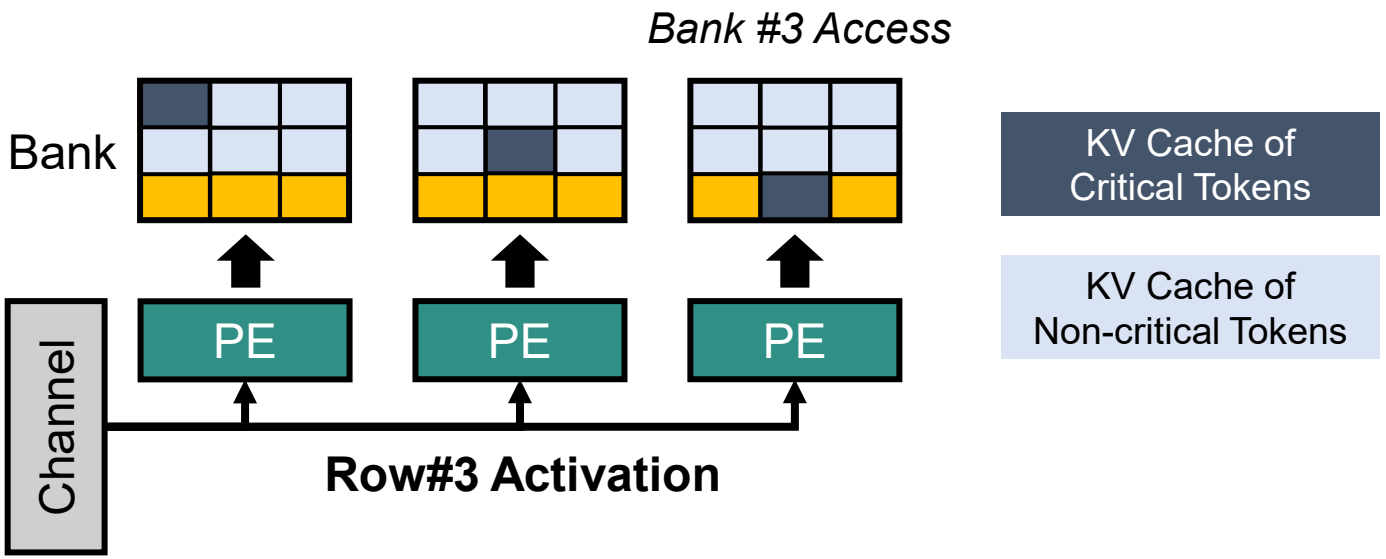


Challenge 1. Data Divergence

- Incurred by different position of operands
- Need access to different row and column addresses
- Requires distinct operation in each bank

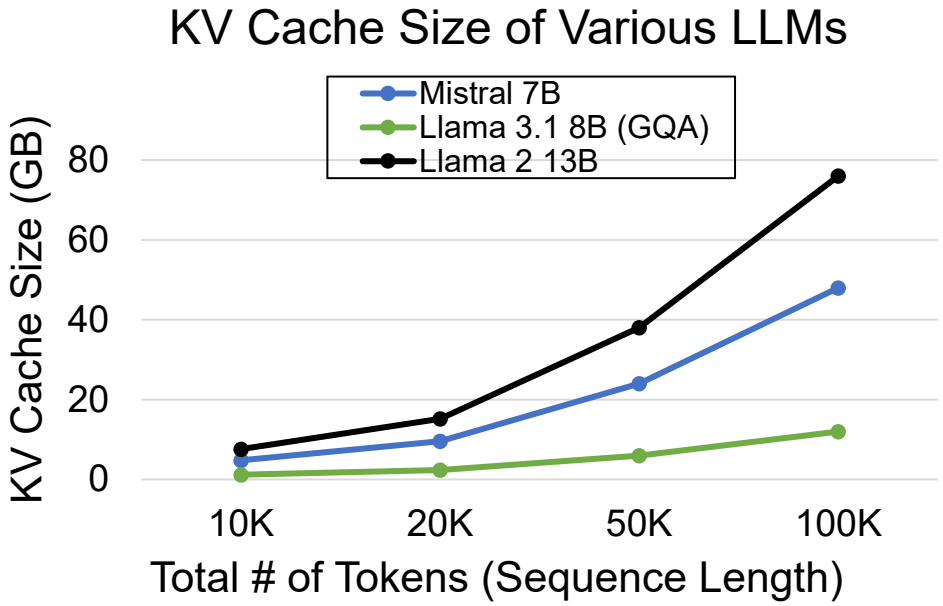
All-bank PIM:

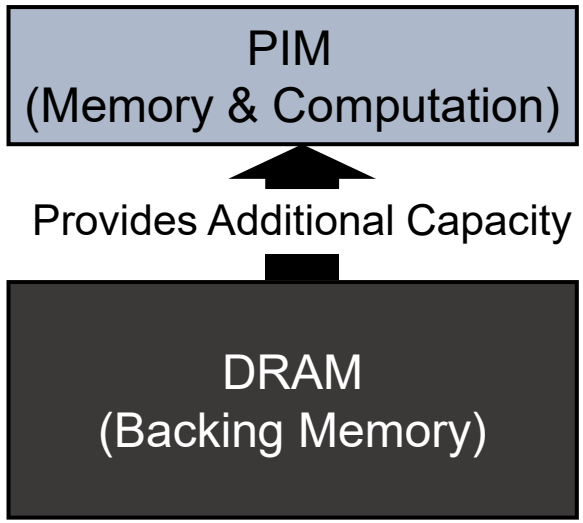
Redundant multiple accesses (row activations)



Challenge 2. Memory Capacity Constraints

- **Dynamic filtering need whole KV cache**
- Small batch & small model: PIM-only
- Large batch & large model: PIM-DRAM





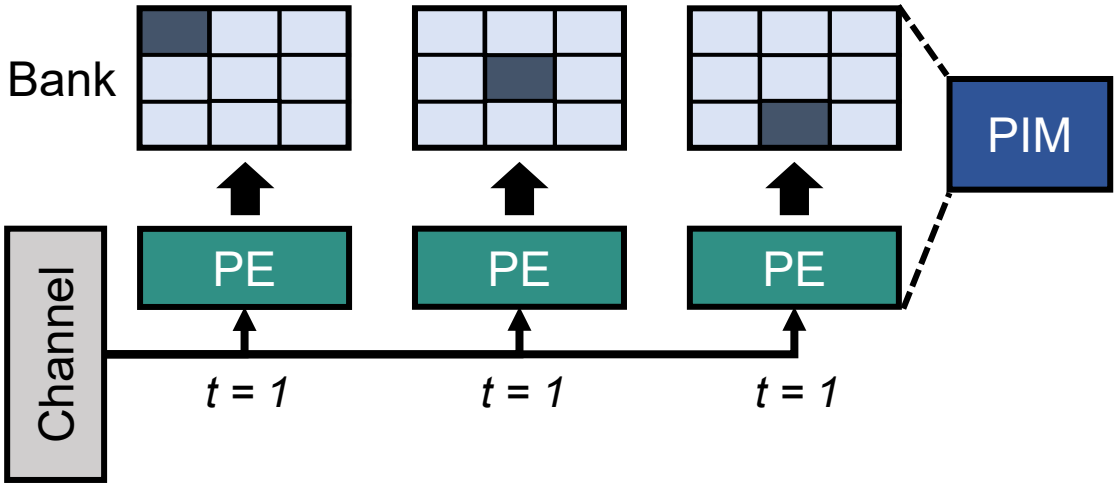
[1] Samsung, “Samsung LPDDR5X-PIM”, Hot Chips 2026.
[2] Lee et al., “Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology”, ISCA 2021.

Contents

- Introduction
- All-bank DRAM PIM
- Challenges
- Design: PASCAL
- Evaluation & Conclusion

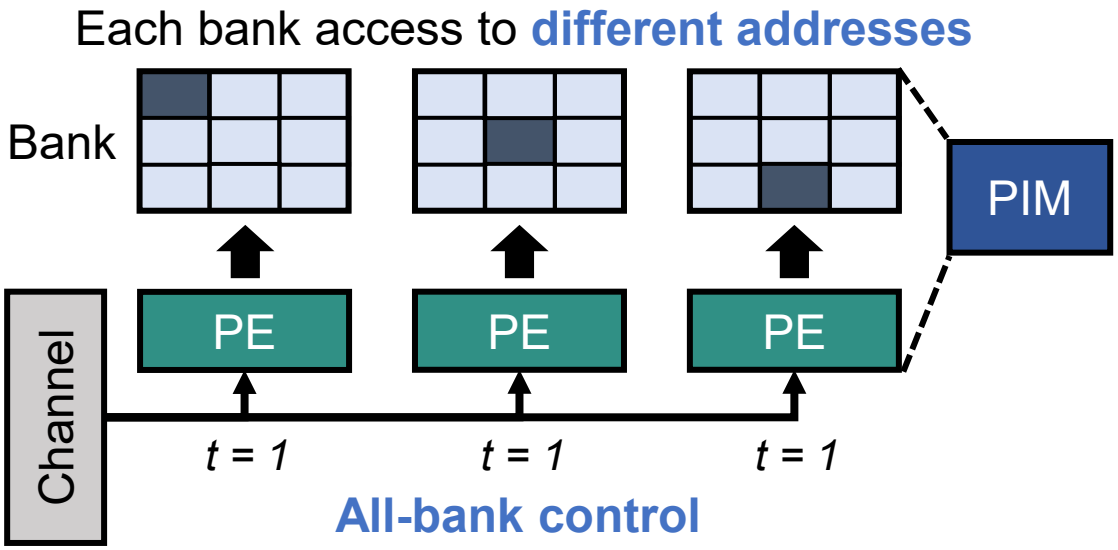
Overview

- Hardware-friendly data layout



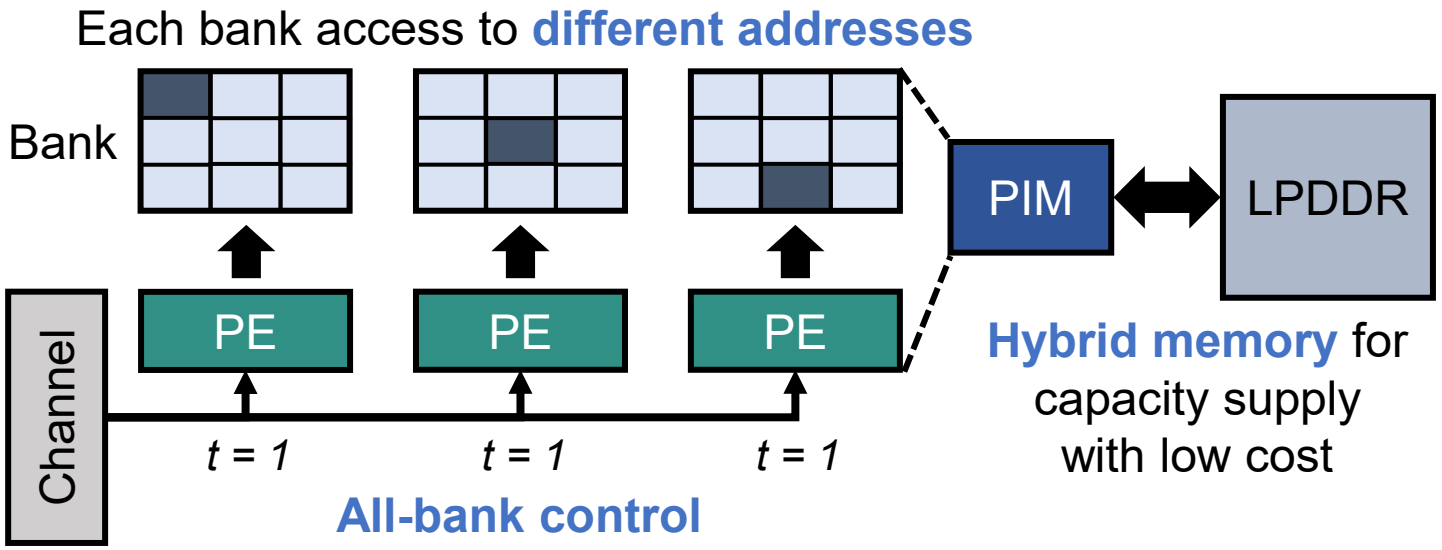
Overview

- Hardware-friendly data layout
- Data divergence handling: HW-based mechanism



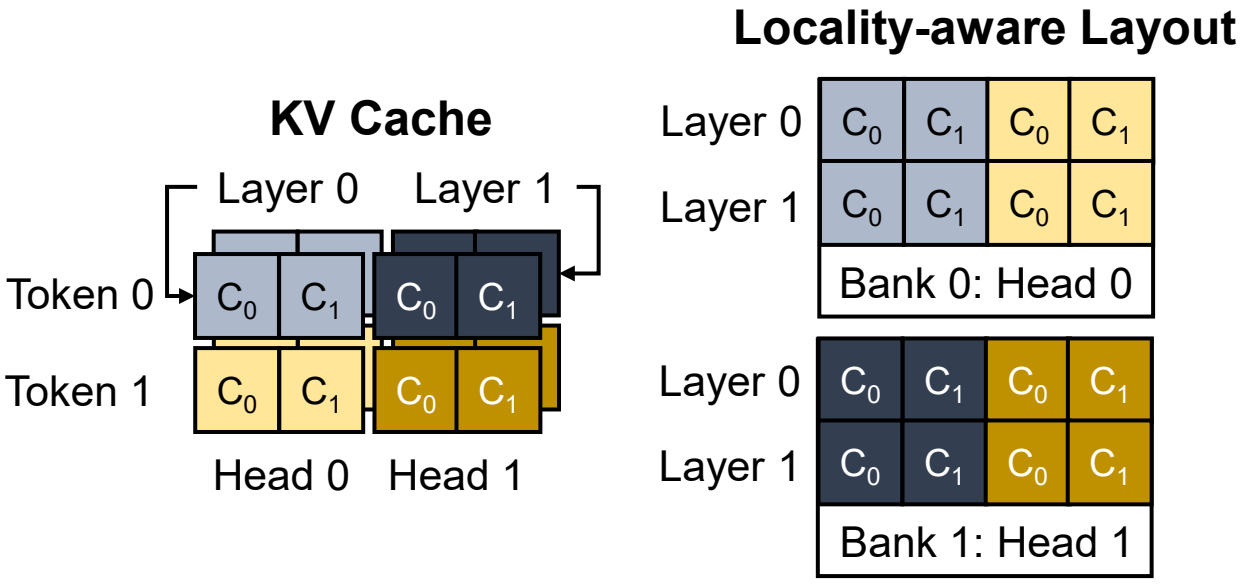
Overview

- Hardware-friendly data layout
- Data divergence handling: HW-based mechanism
- Capacity-aware design: Extension to hybrid memory



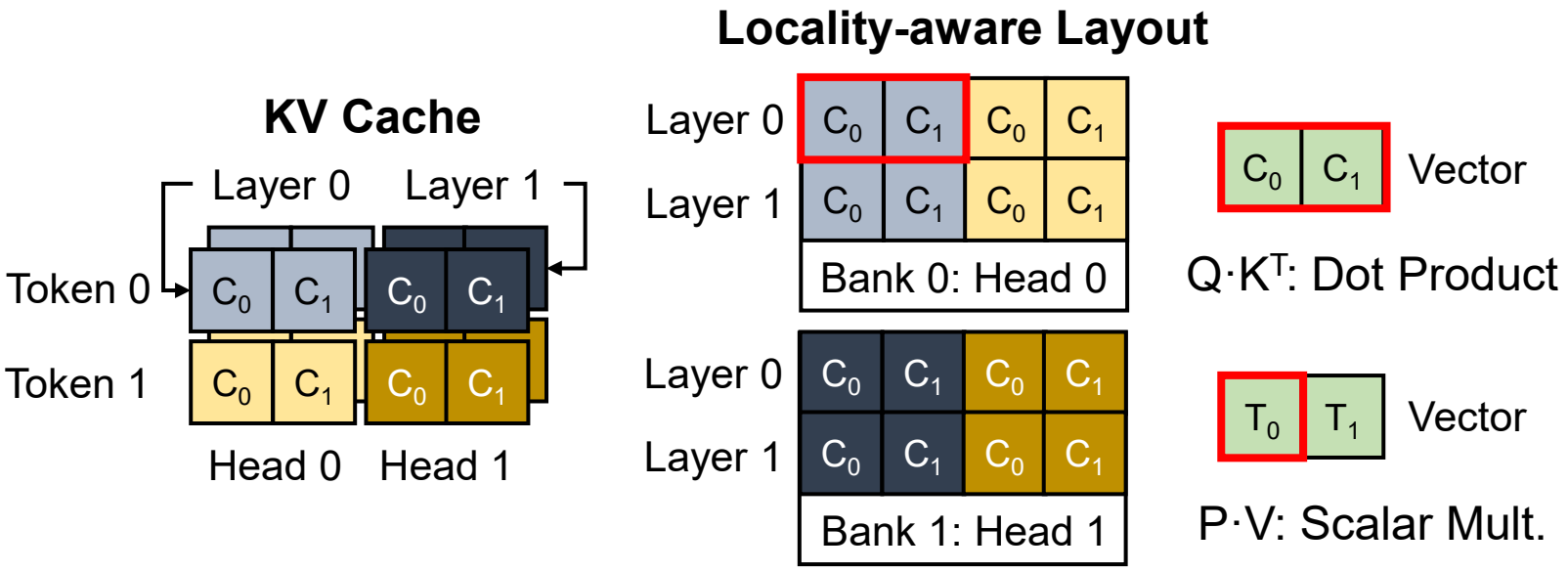
Locality-aware KV Cache Layout

- Unified layout for KV cache
  - Distribute KV cache to banks in head-granularity
  - Row-major: Enables efficient insertion



Locality-aware KV Cache Layout

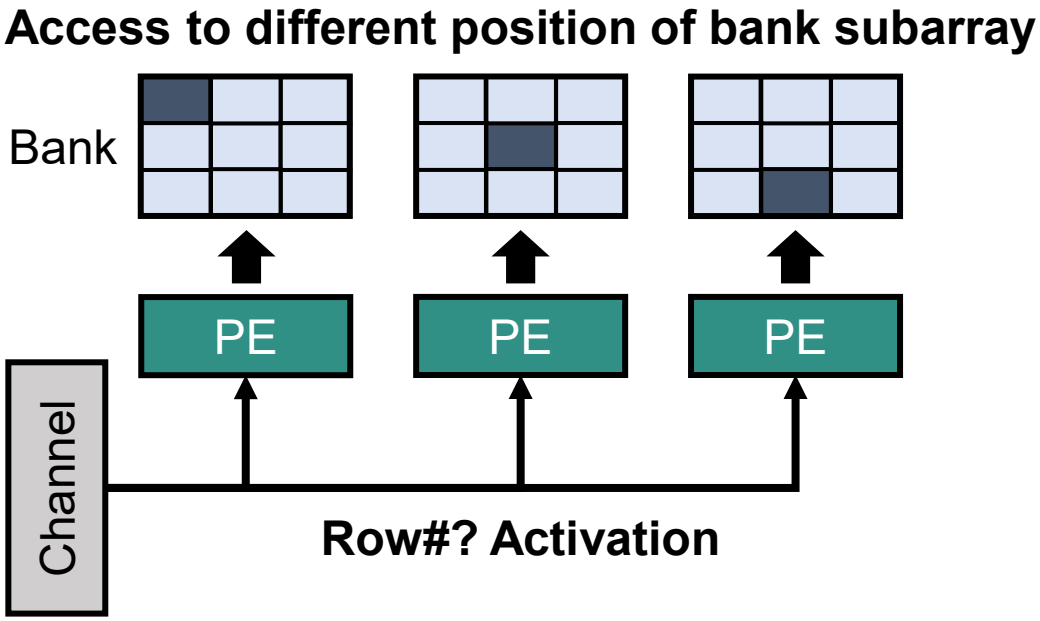
- Unified layout for KV cache
- Algorithm modification for locality
  - $Q \cdot K^T$: Dot Product / $P \cdot V$: Scalar Multiplication
  - Use vector unit with temporal scheduling[1]



[1] Lee et al., "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology", ISCA 2021.

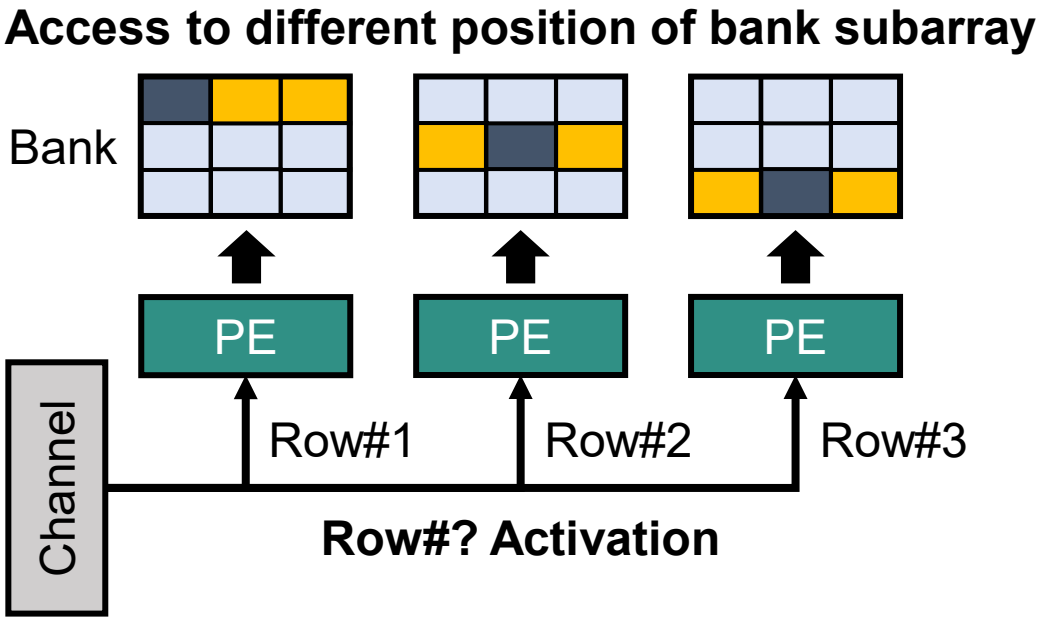
Data Divergence-aware PIM Design

- Goal: Bank-independent control with all-bank PIM



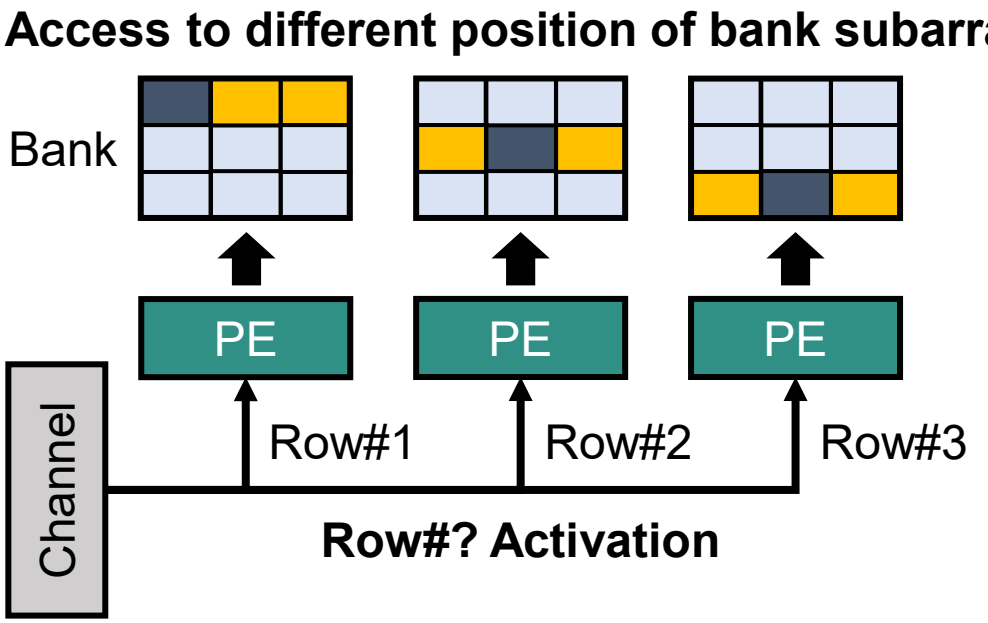
Data Divergence-aware PIM Design

- Goal: Bank-independent control with all-bank PIM



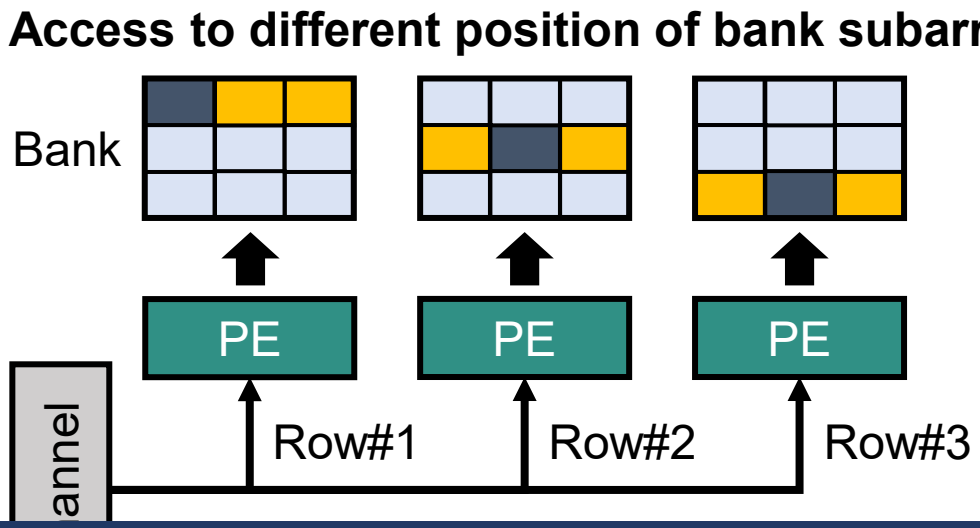
Data Divergence-aware PIM Design

- Goal: Bank-independent control with all-bank PIM
- Hint: Position of critical token (critical token index) is **known before sparse attention computation**



Data Divergence-aware PIM Design

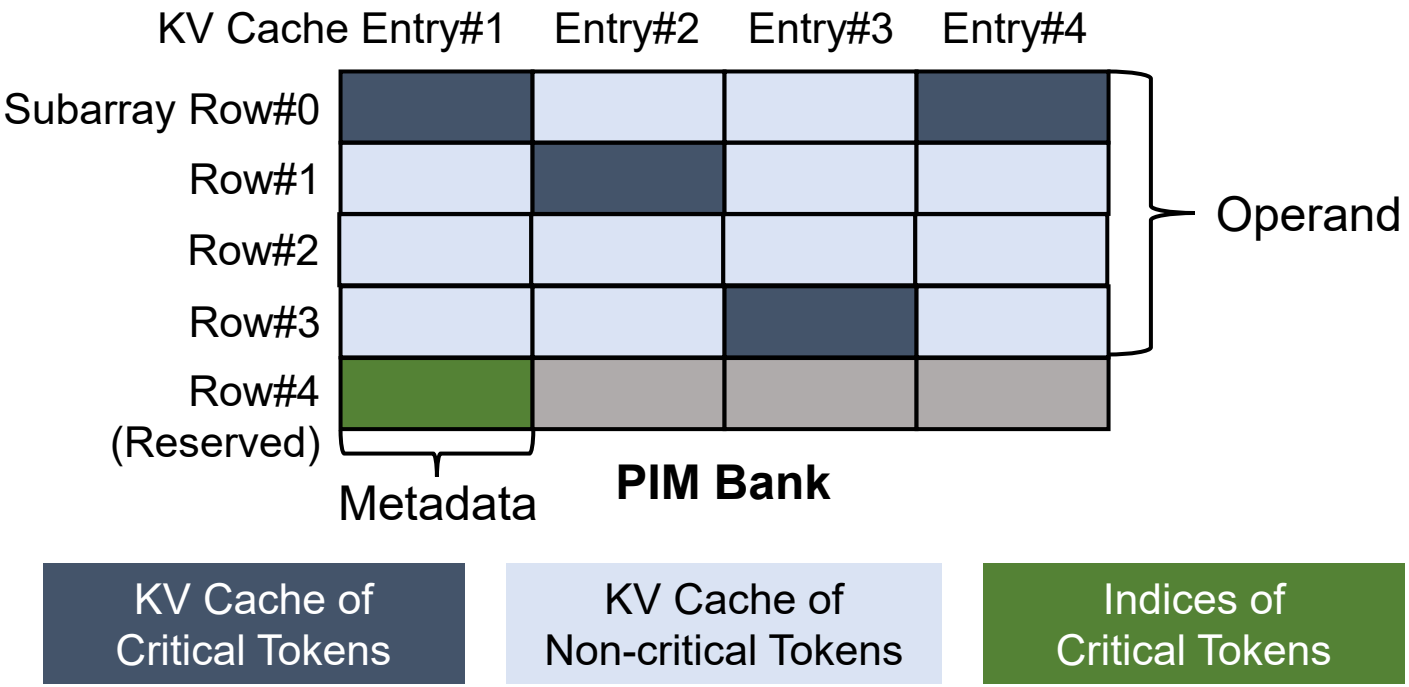
- Goal: Bank-independent control with all-bank PIM
- Hint: Position of critical token (critical token index) is **known before sparse attention computation**



Solution: **Referring critical token index** to compute **absolute position (address)** of operands

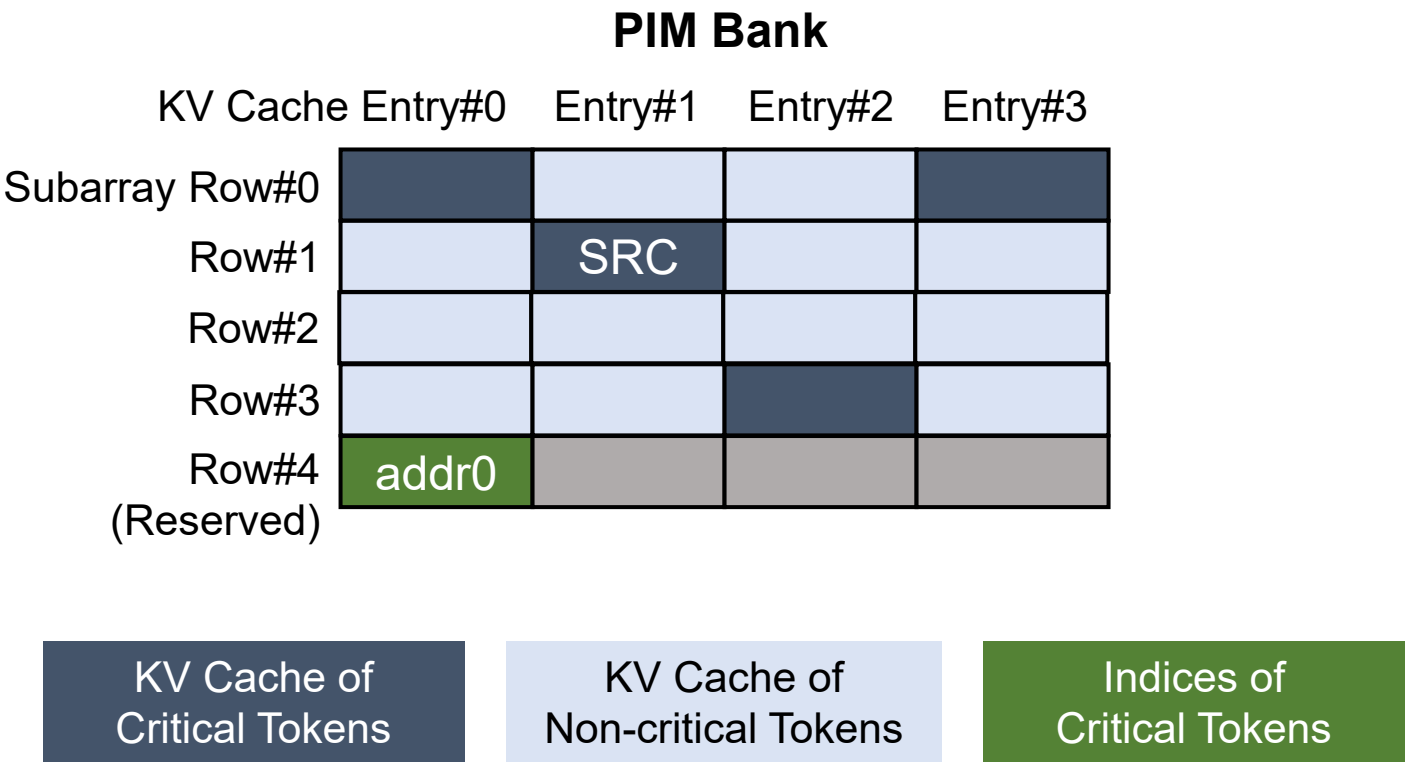
Data Divergence Handling Mechanism

- Bank stores two types of data: Operand and **Metadata**
  - Operand: KV cache entries
  - **Metadata**: Critical token indices
  - Negligible capacity overhead for metadata: **<0.7%**



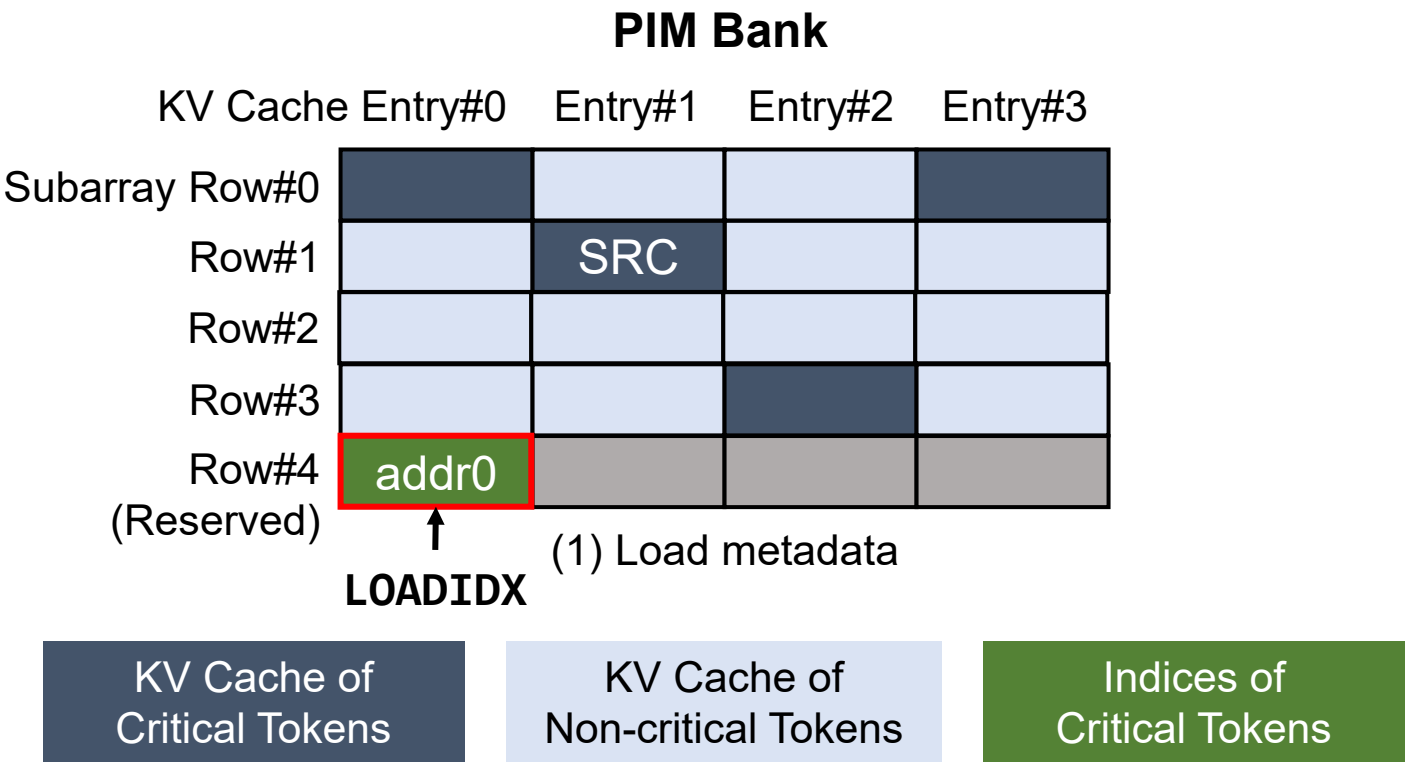
Data Divergence Handling Mechanism

- Activate row by **referring metadata** in bank



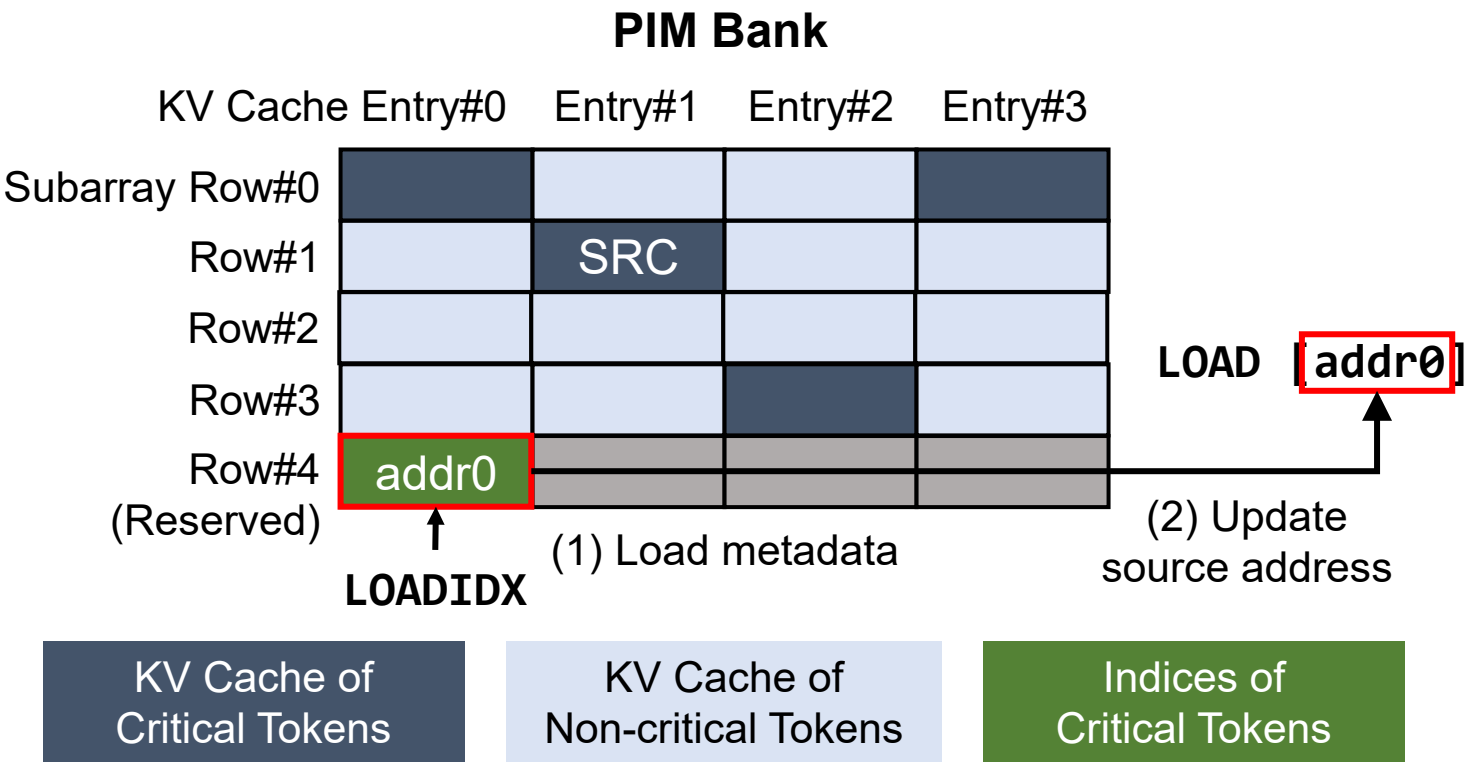
Data Divergence Handling Mechanism

- Activate row by **referring metadata** in bank
  - Modify the **source** of data in data access (LOAD) command



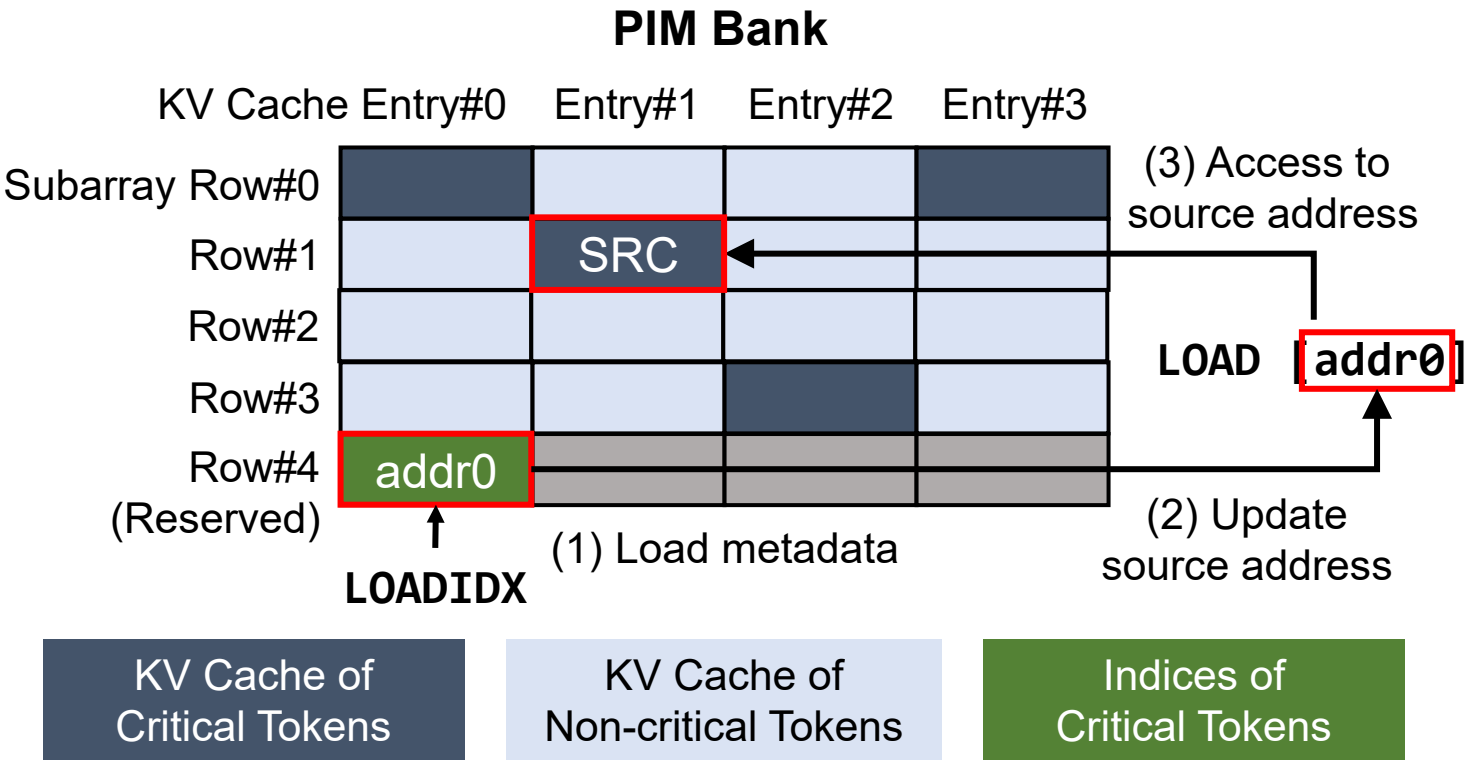
Data Divergence Handling Mechanism

- Activate row by **referring metadata** in bank
  - Modify the **source** of data in data access (LOAD) command



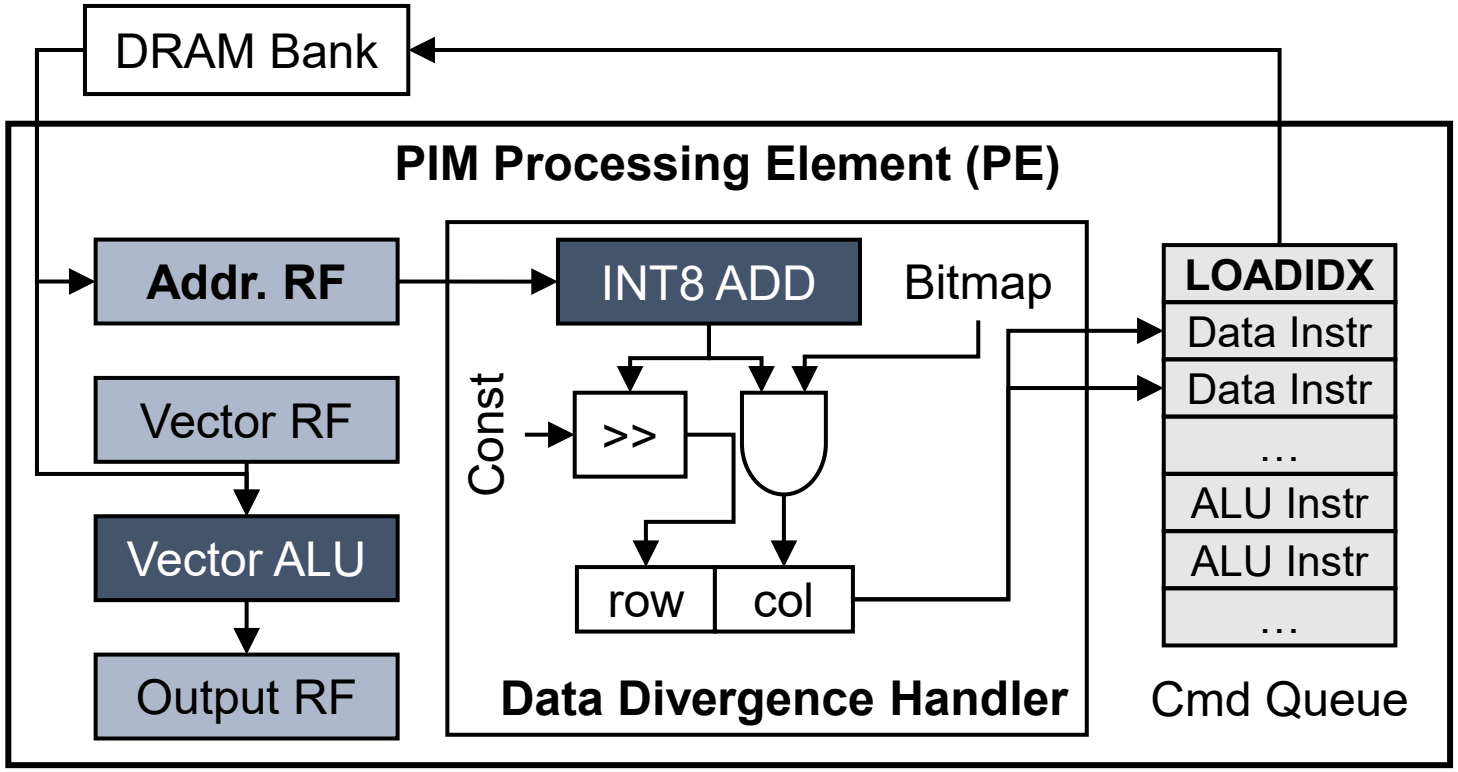
Data Divergence Handling Mechanism

- Activate row by **referring metadata** in bank
  - Modify the **source** of data in data access (LOAD) command
  - Access with updated source address



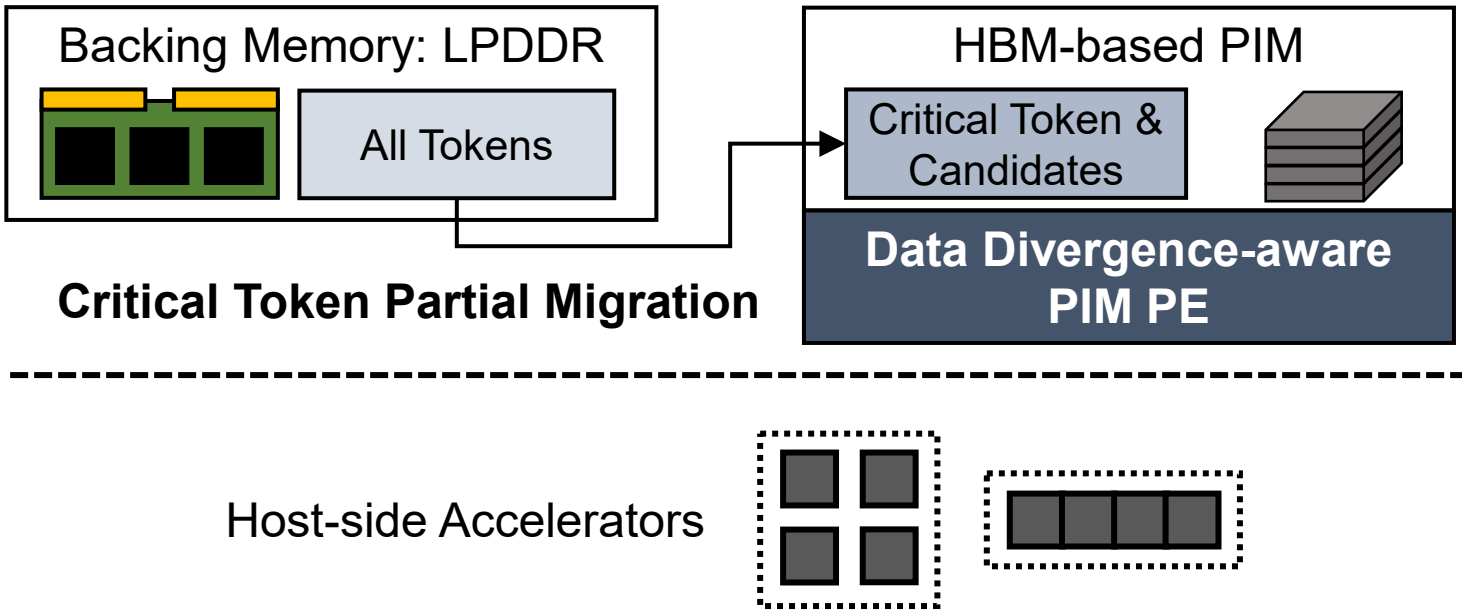
Processing Element Design

- Source address computation with HW component
  - Load metadata with new command **LOADIDX**
  - Only 8% area overhead over the baseline PE design



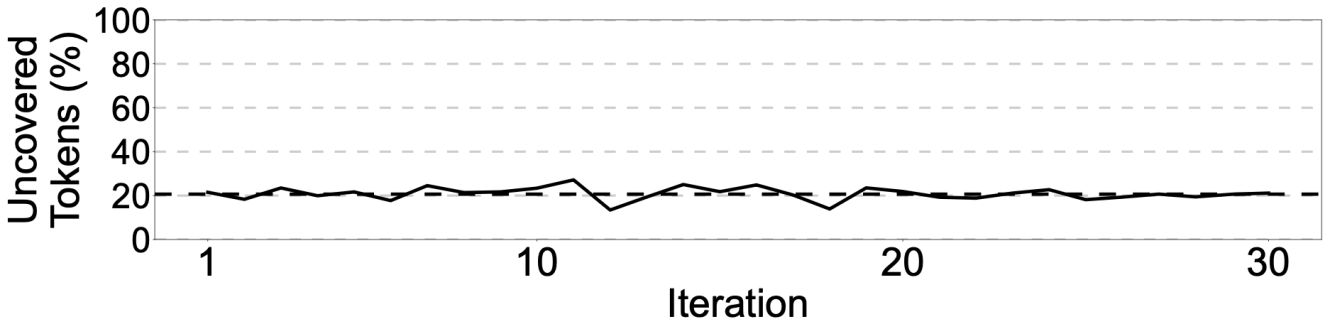
Extension to PIM-DRAM Hybrid System

- LPDDR as a backing memory of HBM PIM
  - LPDDR: Low power/cost, high capacity
- PIM is inclusive to LPDDR to avoid writeback



Critical Token Partial Migration

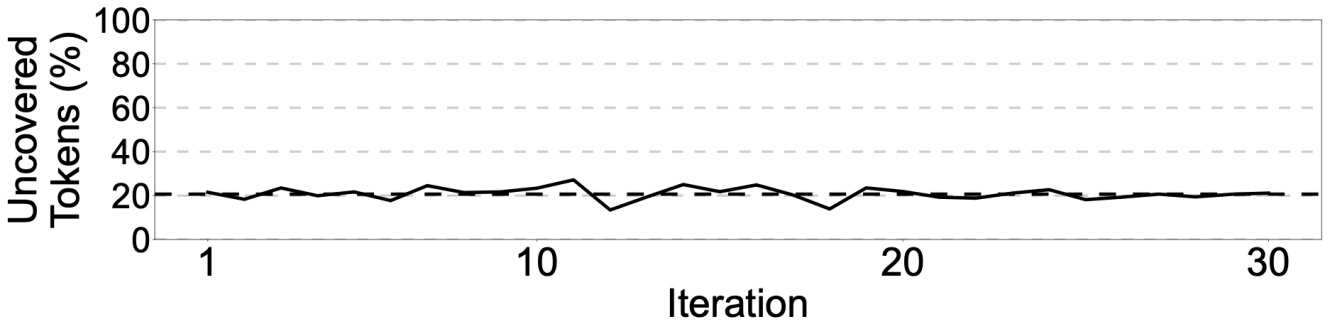
- Exploit locality of critical tokens
  - Only migrates critical tokens not in PIM
  - **Reduces 80% migration data** compared to full migration



Difference of critical token set between iterations: **20.6%** on average

Critical Token Partial Migration

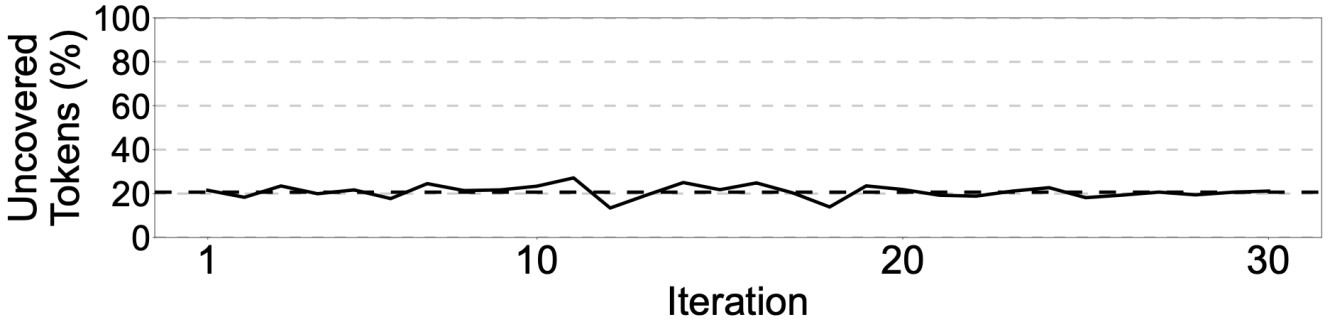
- Exploit locality of critical tokens
  - Only migrates critical tokens not in PIM
  - **Reduces 80% migration data** compared to full migration
- Eviction policy: FIFO
  - FIFO reports only 1.5% increment of miss rate over LRU



Difference of critical token set between iterations: **20.6%** on average

Critical Token Partial Migration

- Exploit locality of critical tokens
  - Only migrates critical tokens not in PIM
  - **Reduces 80% migration data** compared to full migration
- Eviction policy: FIFO
  - FIFO reports only 1.5% increment of miss rate over LRU
- Data divergence-aware PIM is still needed
  - Partial migration does not reduce data divergence



Difference of critical token set between iterations: **20.6%** on average

Contents

- Introduction
- All-bank DRAM PIM
- Challenges
- Design: PASCAL
- Evaluation & Conclusion

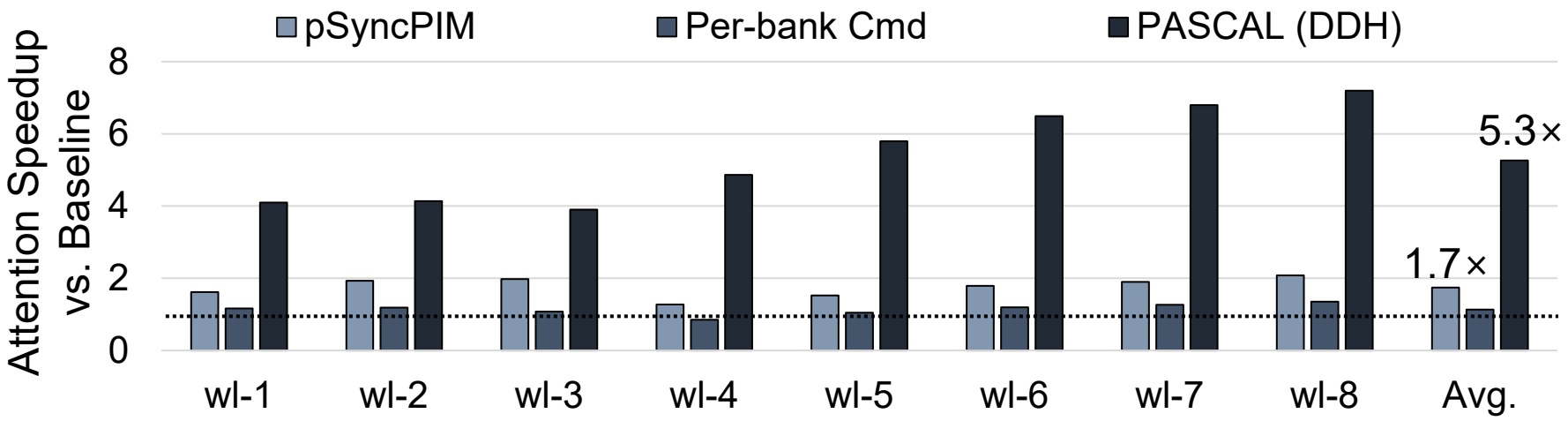
Evaluation Setup

- **12 Sparse LLM workloads** (8 general, 4 hybrid)
  - 3 LLMs: Mistral-7B, Llama3.1-8B, Llama2-13B
  - 2 Filtering schemes: Quest[1], RocketKV[2]
  - Prompt length ranging from 6K to 100K
  - # of critical tokens ranging from 768 to 2048
- System configuration 1: **PIM-only system**
  - HBM2 PIM, 16GB capacity, 1 PE for 4 banks each
- System configuration 2: **PIM-DRAM hybrid system**
  - PIM: HBM2 PIM, 16GB capacity, 1 PE for 4 banks each
  - DRAM: 64GB LPDDR4

[1] Tang et al., “Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference”, ICML 2024.
[2] Behnam et al., “RocketKV: Accelerating Long-Context LLM Inference via Two-Stage KV Cache Compression”, ICML 2025.

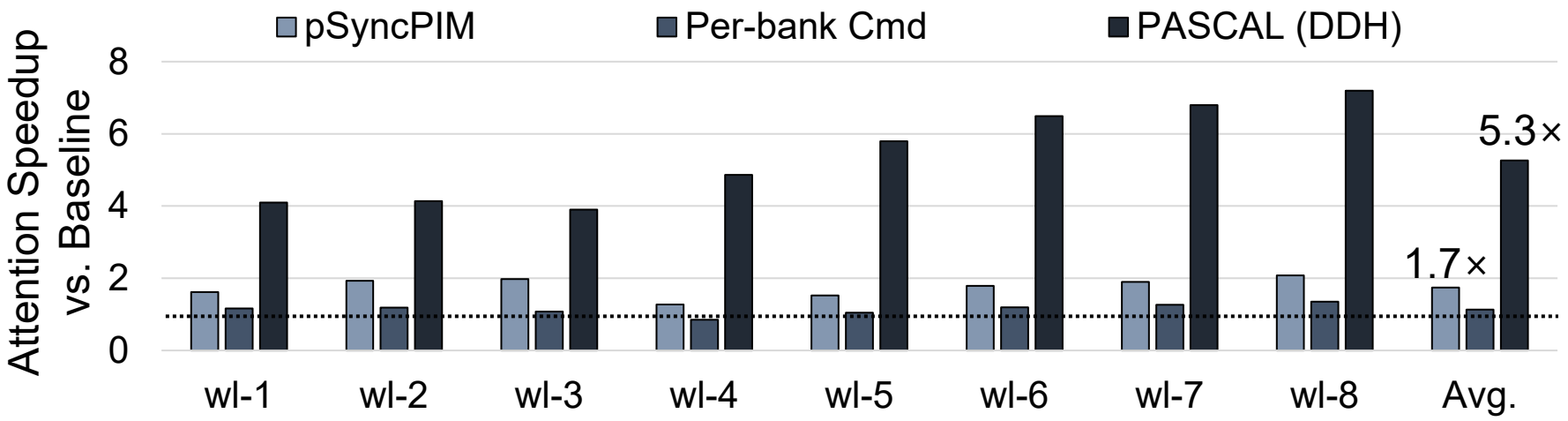
PIM-only System Performance

- Metric: Attention speedup of PIM
  - vs. alternative approaches for handling data divergence
  - Baseline (Normalization point): Dense all-bank PIM



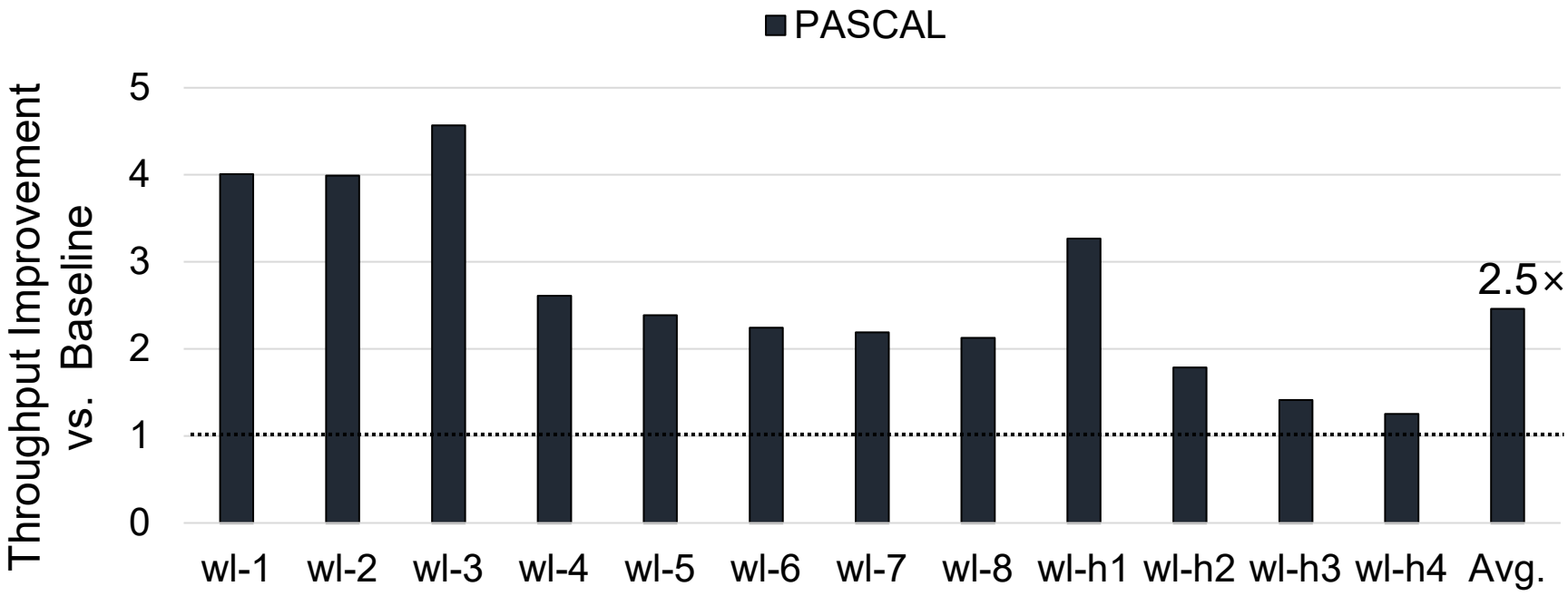
PIM-only System Performance

- Metric: Attention speedup of PIM
  - vs. alternative approaches for handling data divergence
  - Baseline (Normalization point): Dense all-bank PIM
- PASCAL reports **5.3×** speedup over the baseline
  - **3.1×** speedup over prior HPC sparse PIM (pSyncPIM[1])



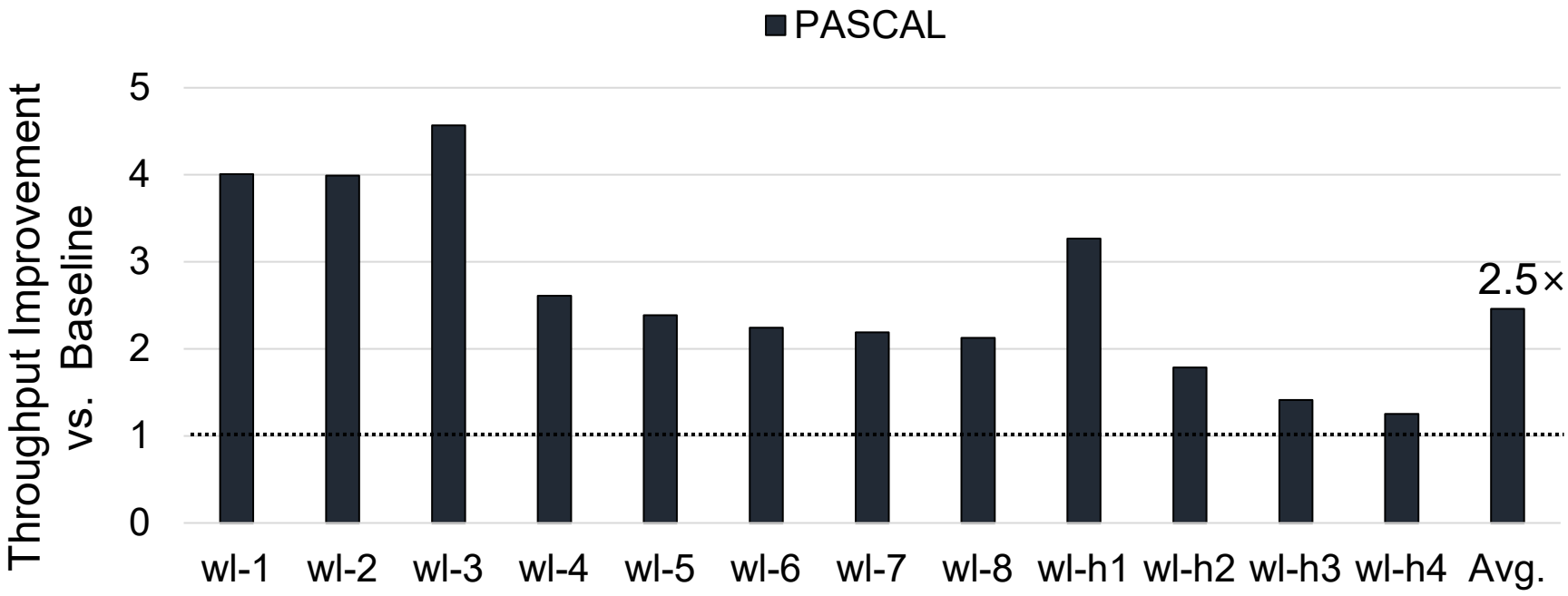
Hybrid System Performance

- Metric: Decoding throughput with NPU-PIM integration
  - Baseline: Dense PIM with full migration of critical tokens
  - Baseline performs dense attention with critical tokens



Hybrid System Performance

- Metric: Decoding throughput with NPU-PIM integration
  - Baseline: Dense PIM with full migration of critical tokens
  - Baseline performs dense attention with critical tokens
- PASCAL reports **2.5×** speedup over the baseline



More Information on Paper

- Detailed PIM Workflow
- Details about PIM Die & Host Configuration
- Performance of PIM-only System with NPU Integration
- Latency Breakdown of PIM-DRAM Hybrid System
- Area and Power Analysis

Conclusion

- Dynamic KV cache filtering: Sparsity & Dynamism
  - **Data divergence** in all-bank PIM
  - Memory **capacity** constraints

Conclusion

- Dynamic KV cache filtering: Sparsity & Dynamism
  - **Data divergence** in all-bank PIM
  - Memory **capacity** constraints

- PASCAL: **Sparse attention + All-bank DRAM PIM**
  - Locality-aware layout for KV cache
  - HW-based data divergence handling mechanism
  - Hybrid memory with efficient migration policy

Conclusion

- Dynamic KV cache filtering: Sparsity & Dynamism
  - **Data divergence** in all-bank PIM
  - Memory **capacity** constraints
- PASCAL: **Sparse attention + All-bank DRAM PIM**
  - Locality-aware layout for KV cache
  - HW-based data divergence handling mechanism
  - Hybrid memory with efficient migration policy
- Highly outperforms conventional all-bank DRAM PIM