COSM: A Cooperative Scheduling Framework for Concurrent PIM and CPU Execution on Mobile Devices

Yilong Zhao(Speaker)*, **Fangxin Liu***, **Onur Mutlu, Mingyu Gao, Jian Liu, Haibing Guan, Li Jiang†**

sjtuzyl@sjtu.edu.cn , liufangxin@sjtu.edu.cn, ljiang_cs@sjtu.edu.cn

Shanghai Jia Tong University | Shanghai Qi Zhi Institute
ETH Zurich | Tsinghua University | Beihang University

ISCA' 2026 July 6, 2026

* *Co-first Author*
† Corresponding Author

Overview

- **Background:** LLM inference with PIM on mobile devices

Overview

- **Background:** LLM inference with PIM on mobile devices
- **Insight: Complementary Bandwidth Occupation**
  CPU/PIM workload: High/Low External BDW; Low/High Internal BDW
- **Goal:** Co-scheduling CPU & PIM to maximize overall bandwidth utilization
- → Cooperative scheduling framework – **Interface + Scheduling**

Overview

- **Background:** LLM inference with PIM on mobile devices
- **Insight: Complementary Bandwidth Occupation**
  CPU/PIM workload: High/Low External BDW; Low/High Internal BDW
- **Goal:** Co-scheduling CPU & PIM to maximize overall bandwidth utilization
- → Cooperative scheduling framework – **Interface + Scheduling**
- **Challenges → Solutions:**

Challenges	Solutions
1 The dilemma of **fixed-length PIM execution command** Interface	*1* **Low-interference Interface** 1.1 Preemptable PIM Execution 1.2 Bandwidth-decoupled Data Transfer
2 **PIM data transfers starve CPU access** under conventional scheduling	
3 Achieve **high CPU & PIM performance**	2 **Idleness-aware Memory Scheduling** 2.1 Idle time window estimate 2.2 Scheduling strategy

Overview

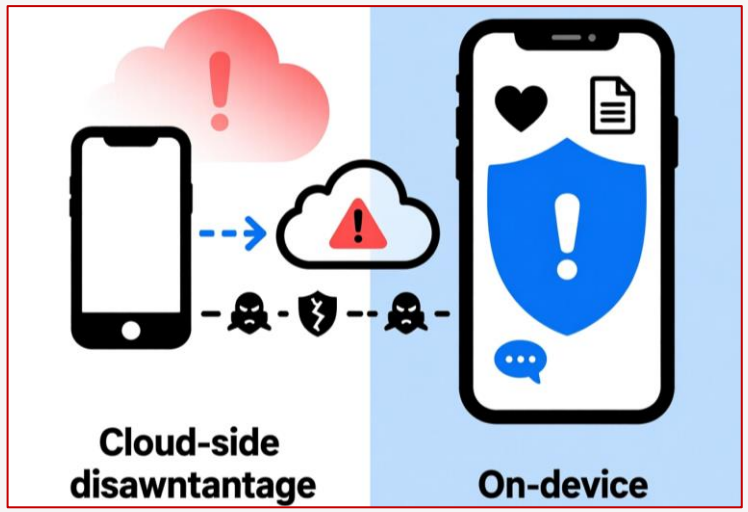
- **Background:** LLM inference with PIM on mobile devices
- **Insight: Complementary Bandwidth Occupation**
  CPU/PIM workload: High/Low External BDW; Low/High Internal BDW
- **Goal:** Co-scheduling CPU & PIM to maximize overall bandwidth utilization
- → Cooperative scheduling framework – **Interface + Scheduling**
- **Challenges → Solutions:**

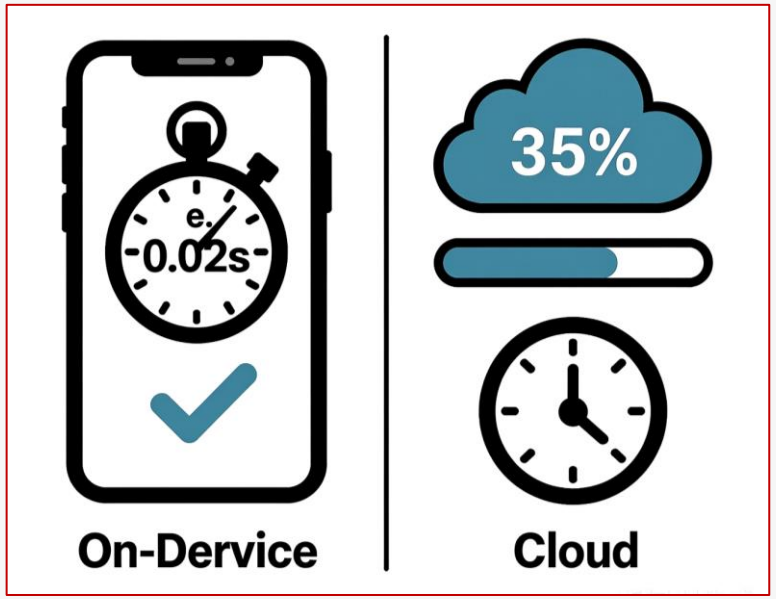
Challenges	Solutions
1 The dilemma of **fixed-length PIM execution command** Interface	*1* **Low-interference Interface** 1.1 Preemptable PIM Execution 1.2 Bandwidth-decoupled Data Transfer
2 **PIM data transfers starve CPU access** under conventional scheduling	
3 Achieve **high CPU & PIM performance**	2 **Idleness-aware Memory Scheduling** 2.1 Idle time window estimate 2.2 Scheduling strategy

- **Evaluation:** Ramulator2 simulation | 2-Channel LPDDR5 | Mobile APP + LLM
- **Results:** 2.8x PIM Throughput with only 2% CPU Degradation.

LLM Inference on Mobile

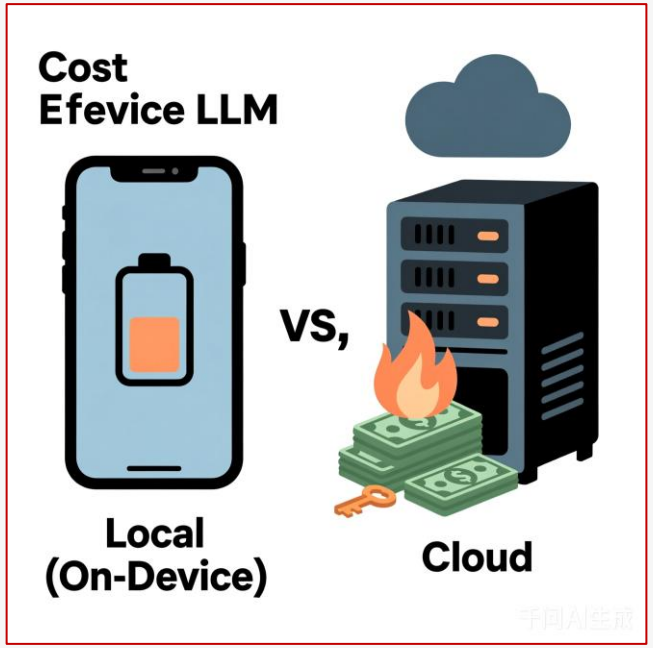
Privacy & Data Sovereignty





Real-time Response

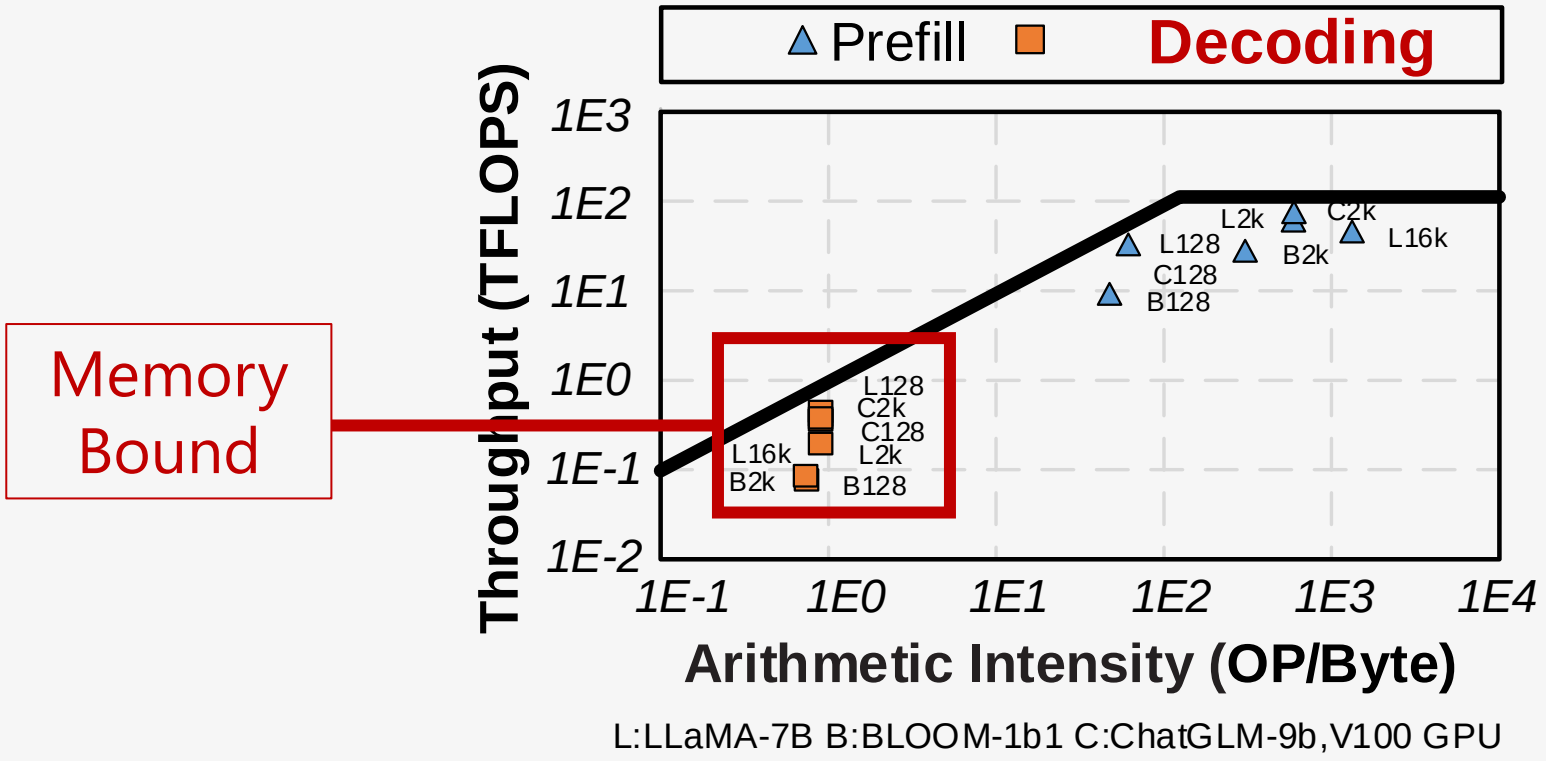
Cost Efficiency & Scalability



These Requirements Demand **Mobile LLM Inference**

LLM Decoding is Memory Bound

Limited resource on mobile → Batch size =1 → Decoding is memory bound



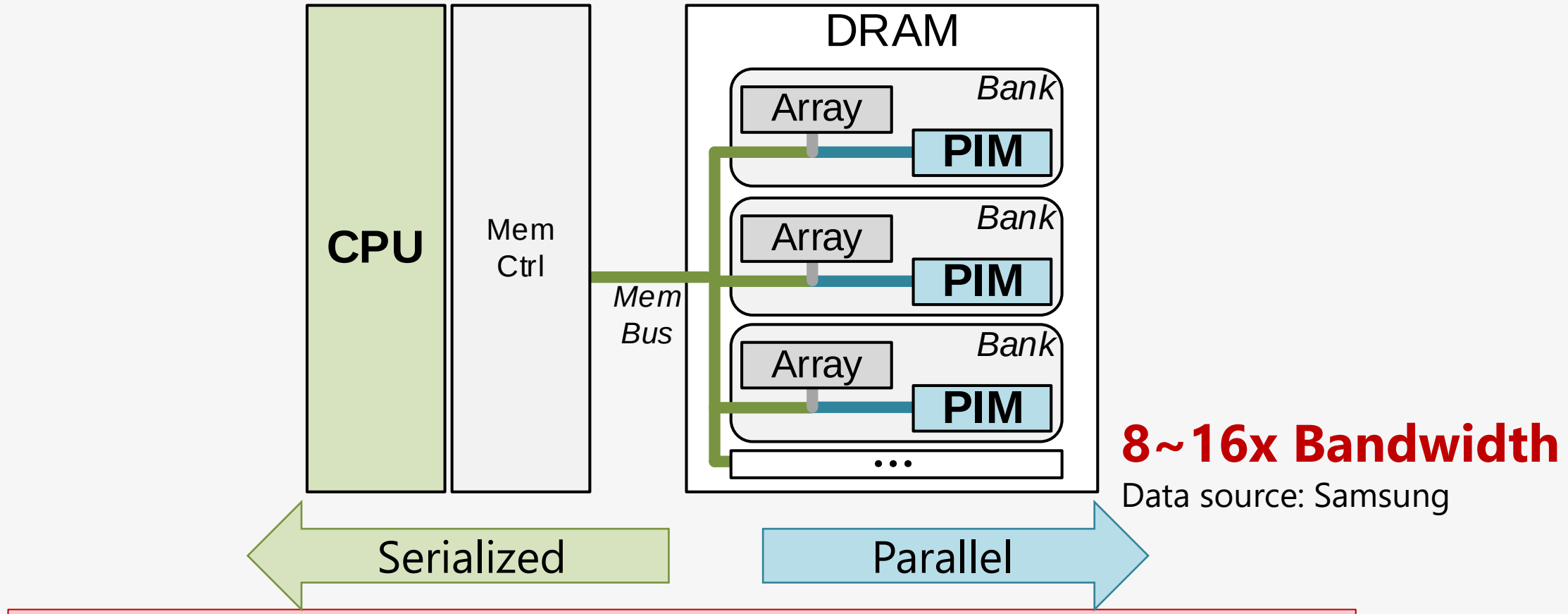
L:LLaMA-7B B:BLOOM-1b1 C:ChatGLM-9b,V100 GPU

>80% time
>70% energy
on data transfer
Samsung [HCS'23]

LLM decoding is inefficient on NPU/GPU because of <u>limited bandwidth</u>!

DRAM-based Processing-in-Memory (PIM)

PIM: Integrate Computing Units in DRAM Banks



8~16x Bandwidth

Data source: Samsung

PIM achieve high bandwidth through parallel access

CPU-PIM Memory Contention

Mobile device faces **limited resource**

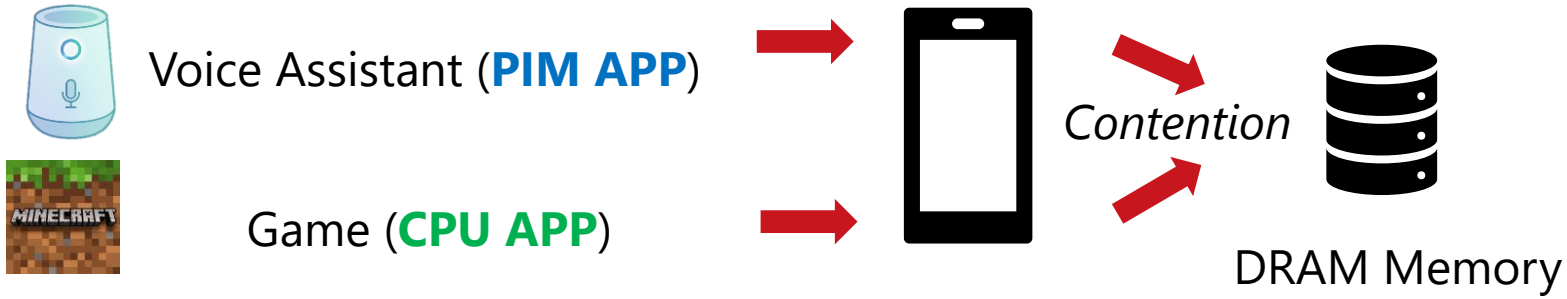
→ CPU & PIM should **share memory**

CPU-PIM Memory Contention

Mobile device faces **limited resource**

→ CPU & PIM should **share memory**

CPU & PIM face **memory access contention**



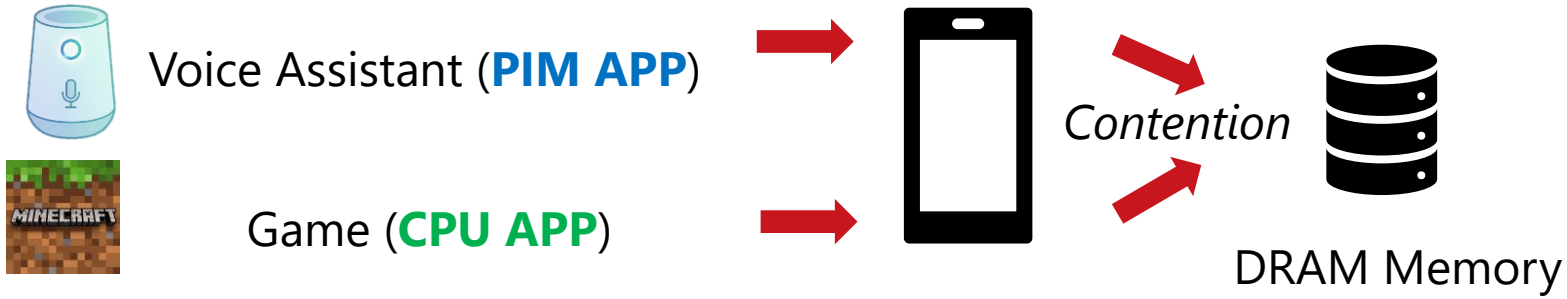
Significant performance degradation is unacceptable

CPU-PIM Memory Contention

Mobile device faces **limited resource**

→ CPU & PIM should **share memory**

CPU & PIM face **memory access contention**

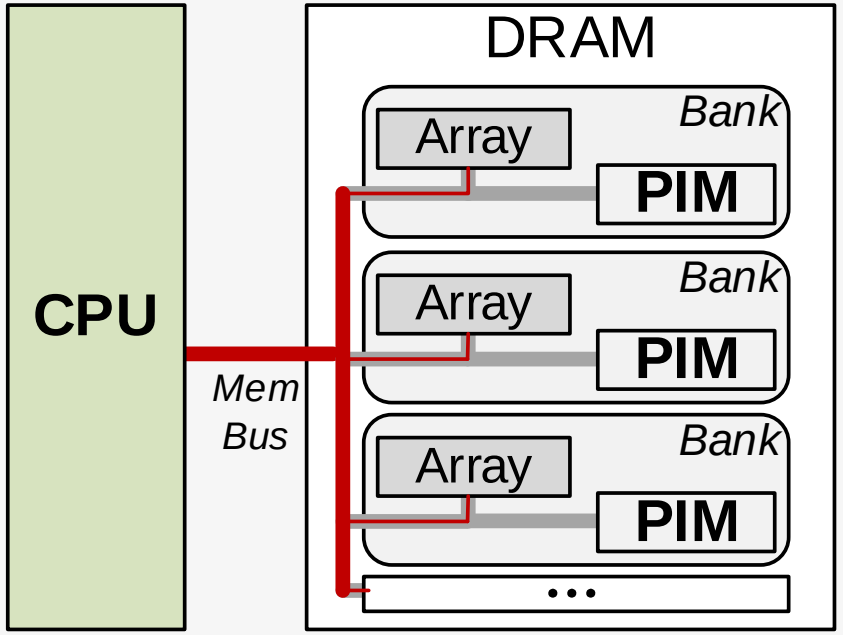


Significant performance degradation is unacceptable

How to **maximize the utilization** of available **bandwidth** resources?

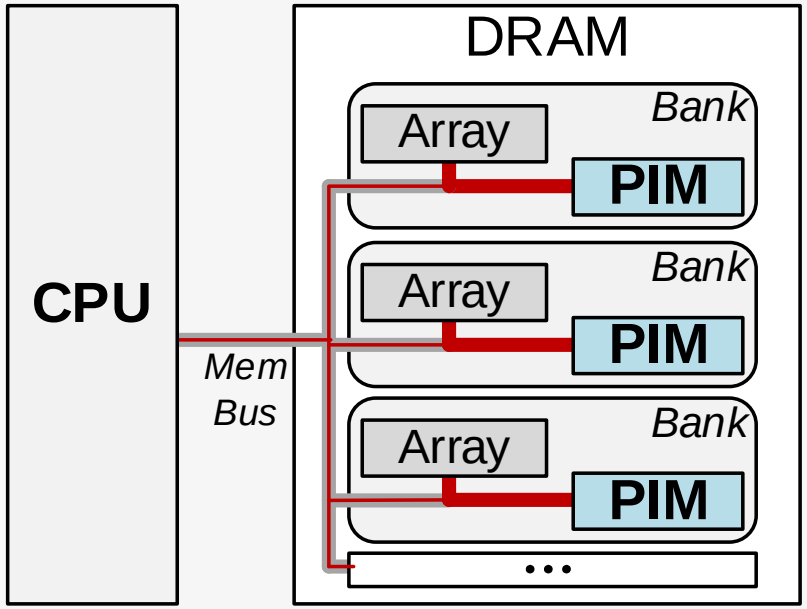
PIM & CPU Workload: Inefficient Bandwidth Use

External bandwidth: memory bus bandwidth | ***Internal bandwidth***: aggregated bank bandwidth



CPU Workload:
Only accesses

High External Bandwidth ↑
Low Internal Bandwidth ↓

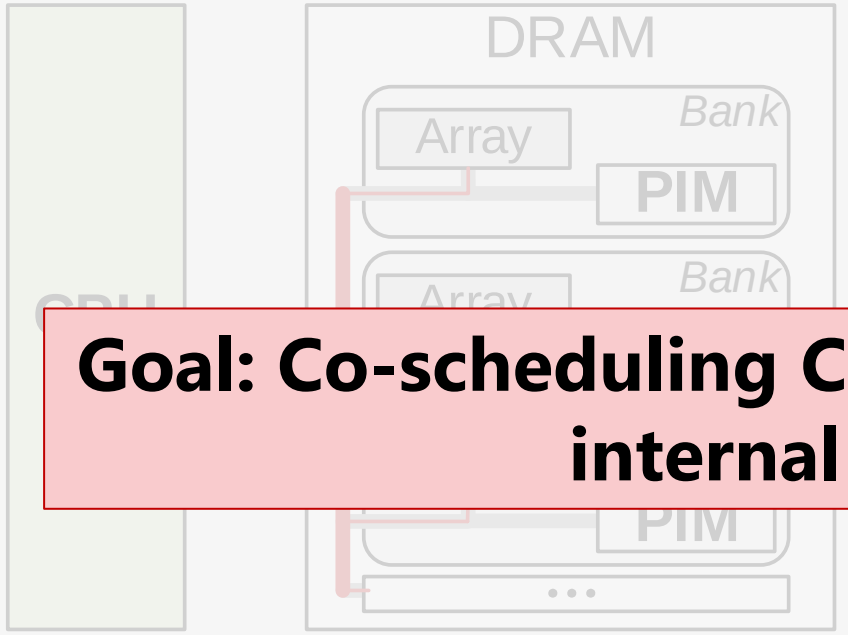


PIM Workload:
A few accesses + Massive executions

Low External Bandwidth ↓
High Internal Bandwidth ↑

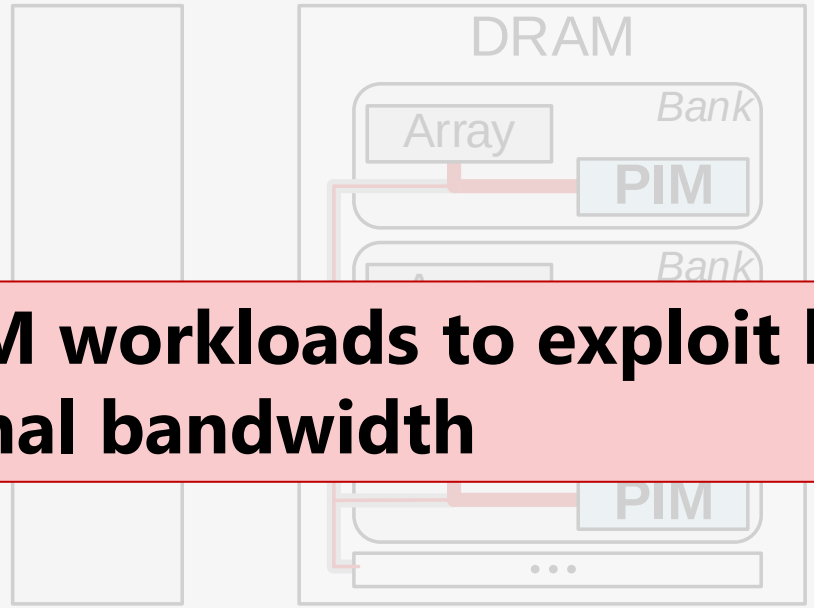
PIM & CPU Workload: Inefficient Bandwidth Use

External bandwidth: memory bus bandwidth | ***Internal bandwidth***: aggregated bank bandwidth



CPU Workload:
Only accesses

High **External Bandwidth** ↑
Low **Internal Bandwidth** ↓



PIM Workload:
A few accesses + Massive executions

Low **External Bandwidth** ↓
High **Internal Bandwidth** ↑

Goal: Co-scheduling CPU & PIM workloads to exploit both internal & external bandwidth

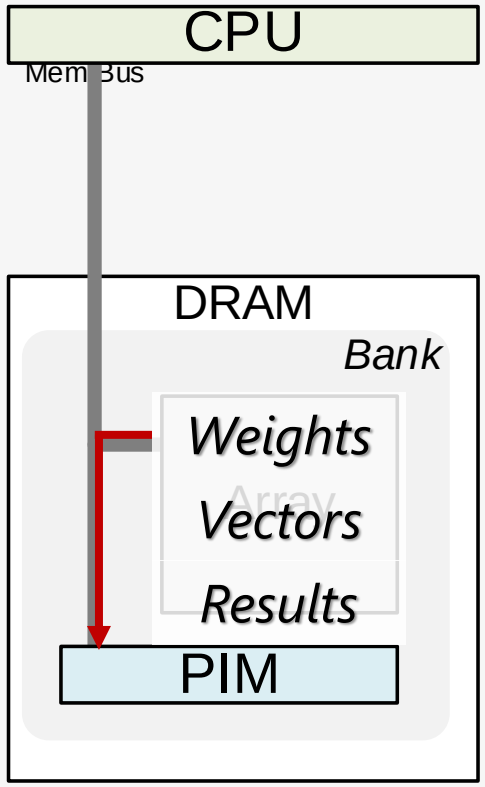
PIM Command Support

PIM units can only access local bank

PIM Execution

via

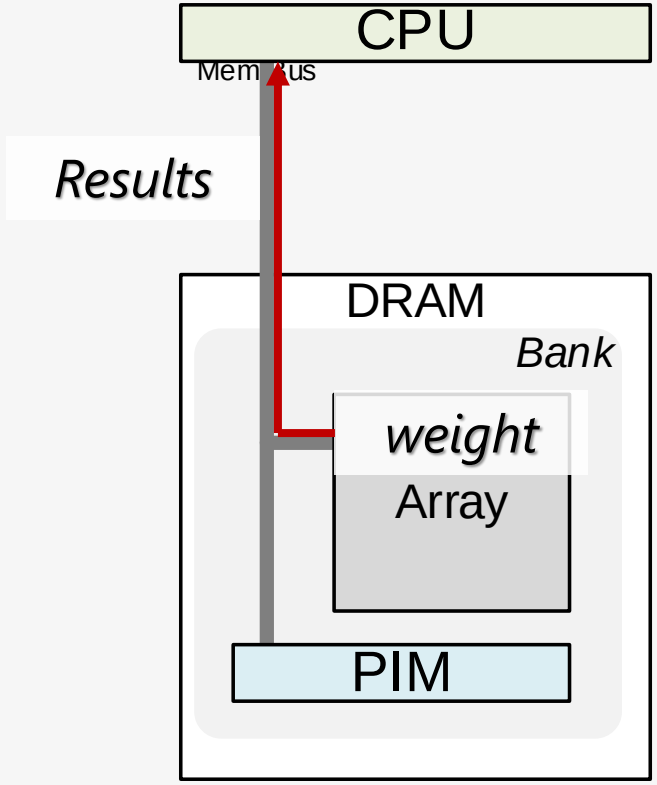
extended DRAM command



PIM Data Transfer (between Banks)

via

normal read/write command



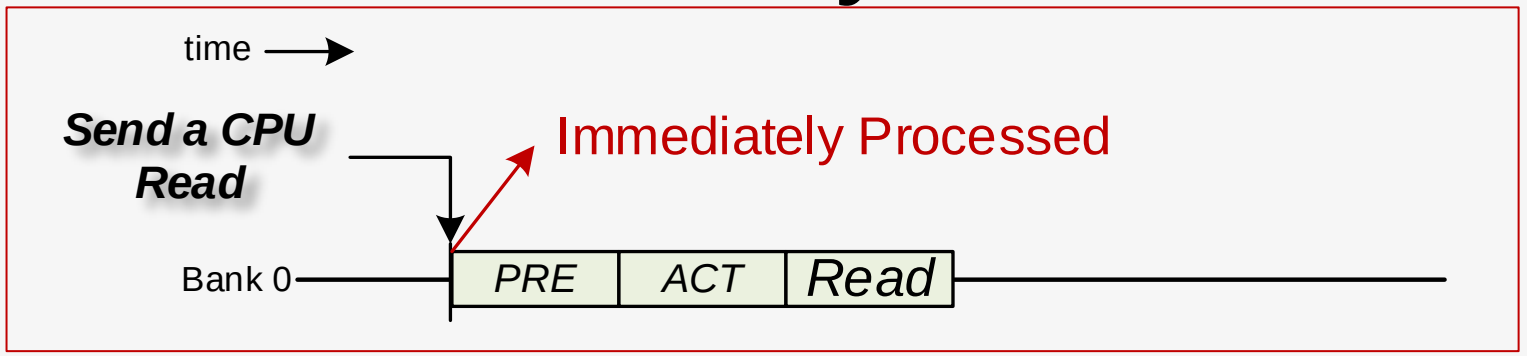
Interface Analysis I:

Interference between **PIM Execution** & **CPU Access**

Observation 1

PIM interference on CPU access latency:

CPU access only:



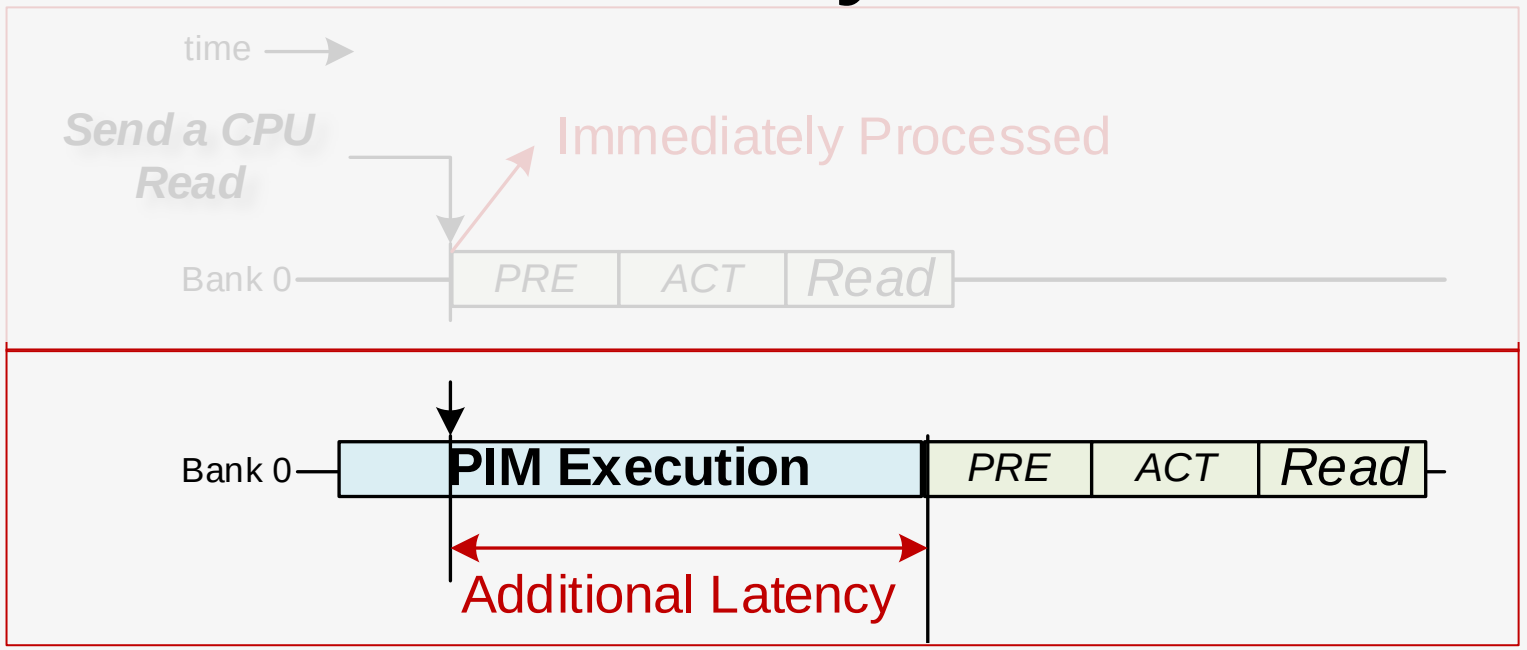
Define ***Command Length***: # Cycles of PIM Execution

Observation 1

PIM interference on CPU access latency:

CPU access only:

With PIM execution:

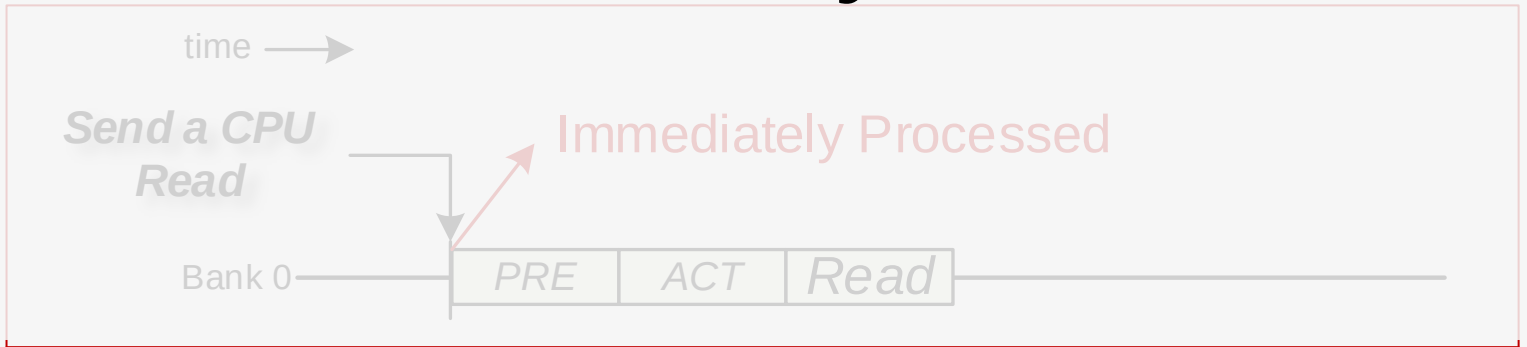


Define ***Command Length***: # Cycles of PIM Execution

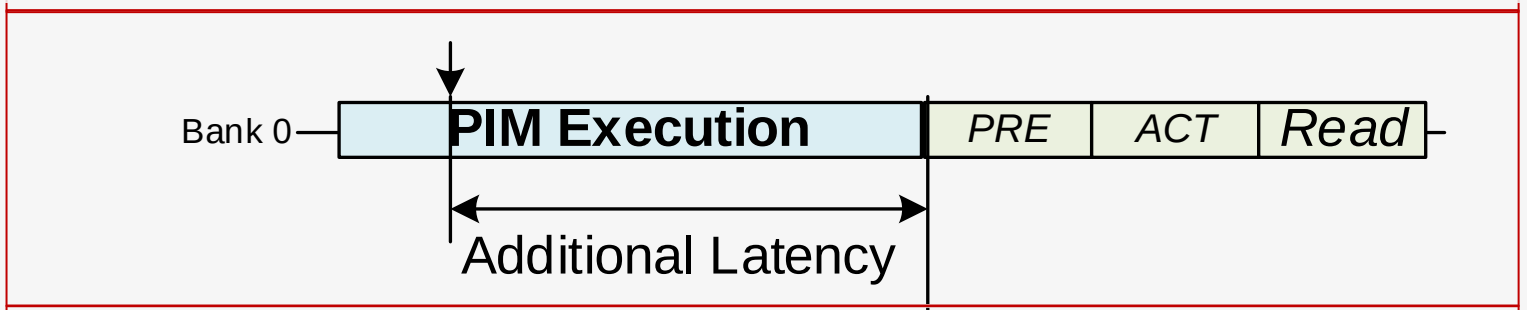
Observation 1

PIM interference on CPU access latency:

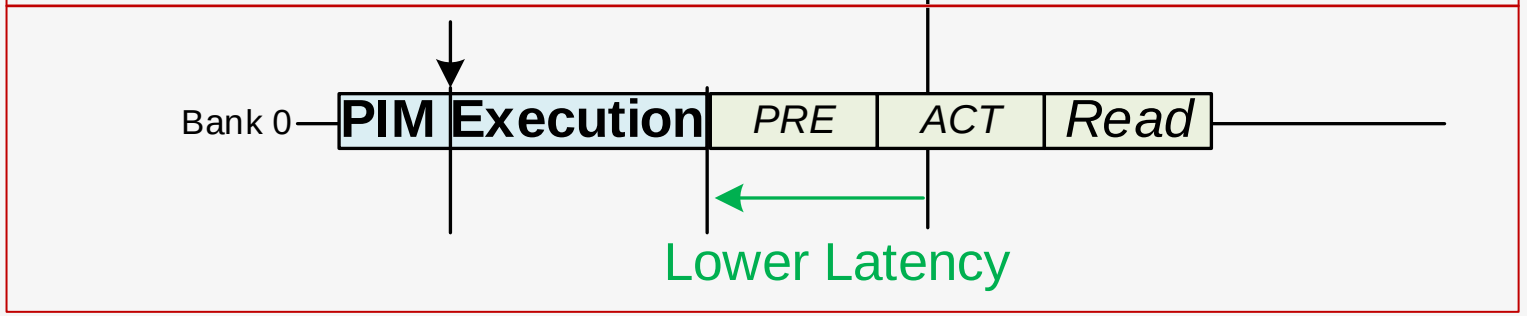
CPU access only:



With PIM execution:



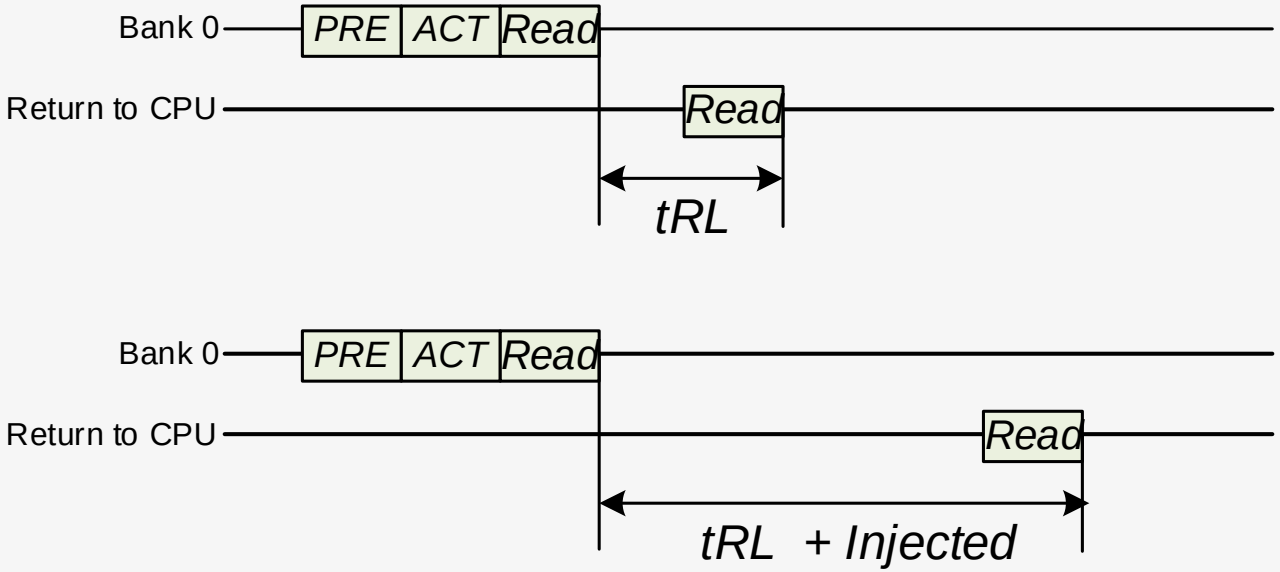
With a ***Shorter*** PIM Execution:

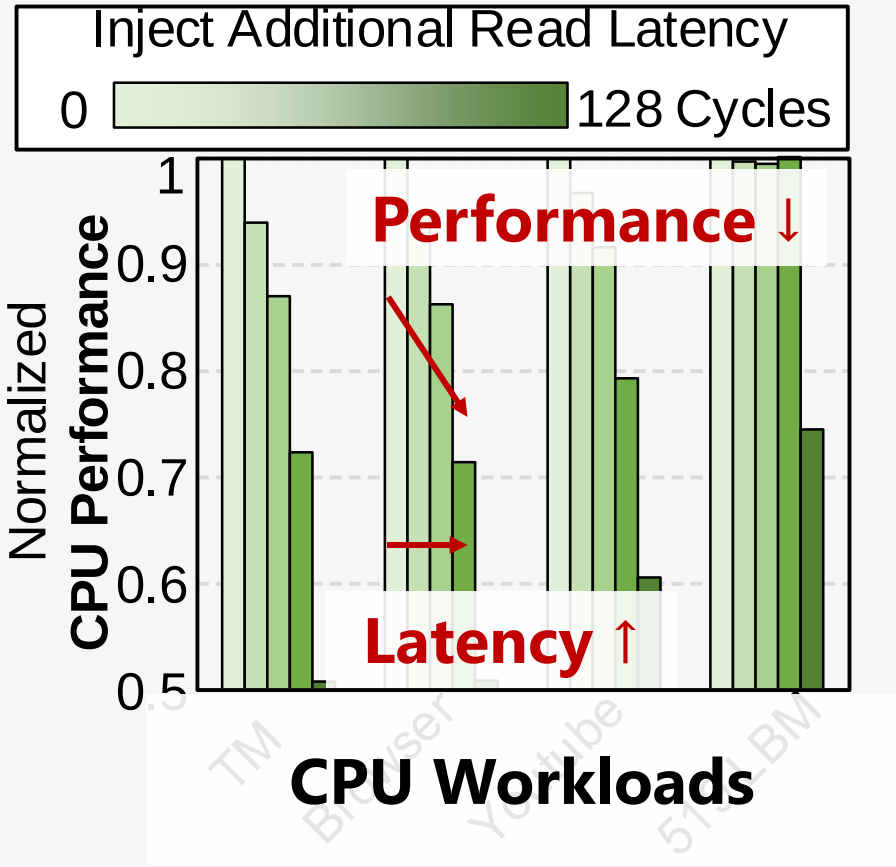


Define ***Command Length***: # Cycles of PIM Execution

Observation 1

Experimental Setup:
Inject Additional Read Latency in Ramulator2 Simulator





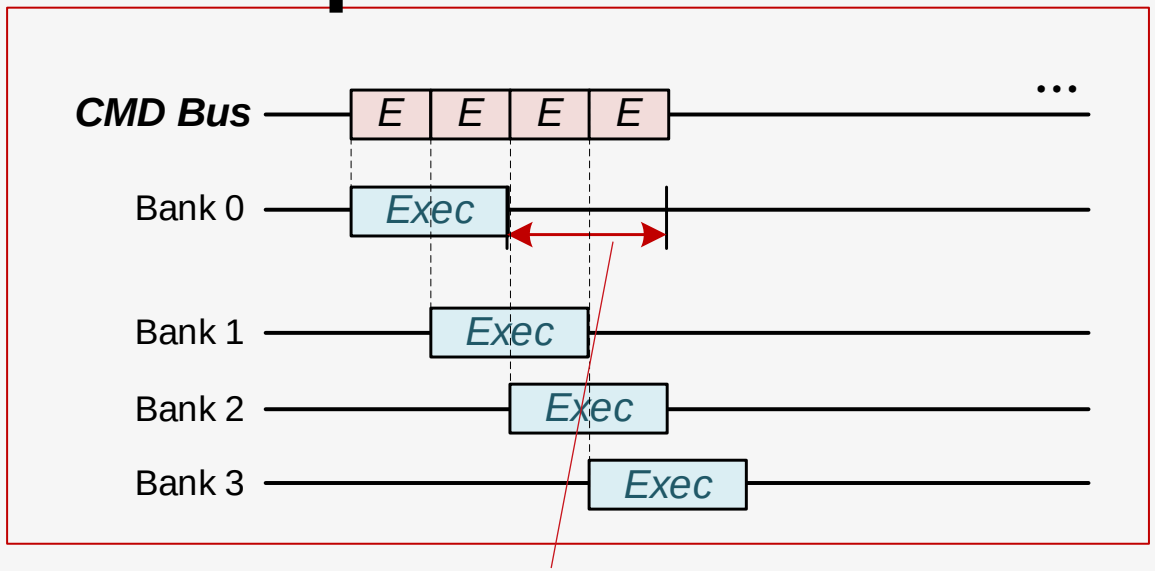
CPU workload is sensitive to read latency

PIM execution should be Short

Observation 2

PIM underutilization due to bus pressure:

Short PIM execution command:

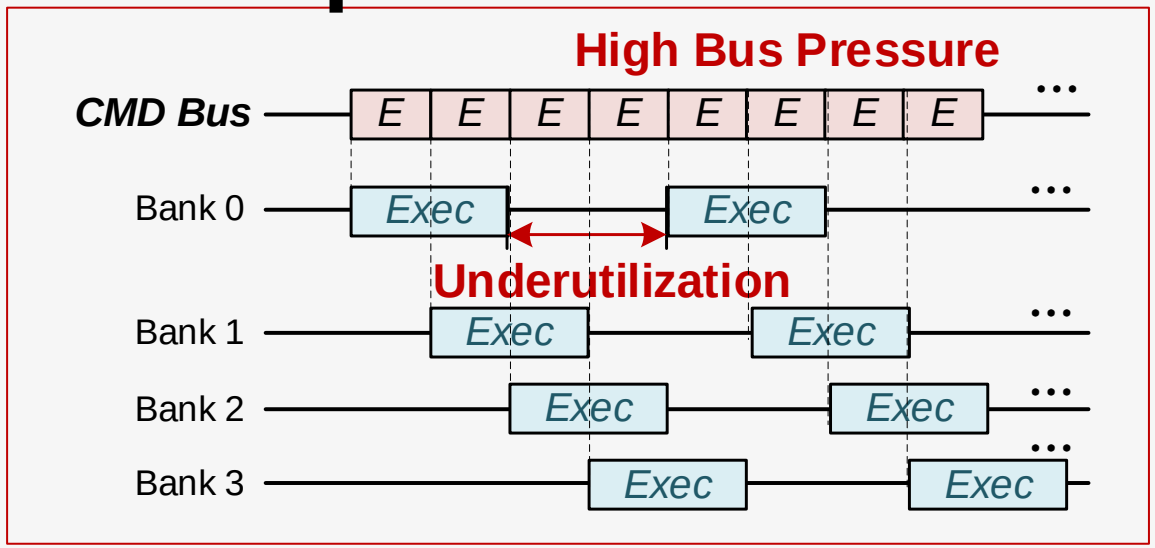


Idle before rounded back

Observation 2

PIM underutilization due to bus pressure:

Short PIM execution command:

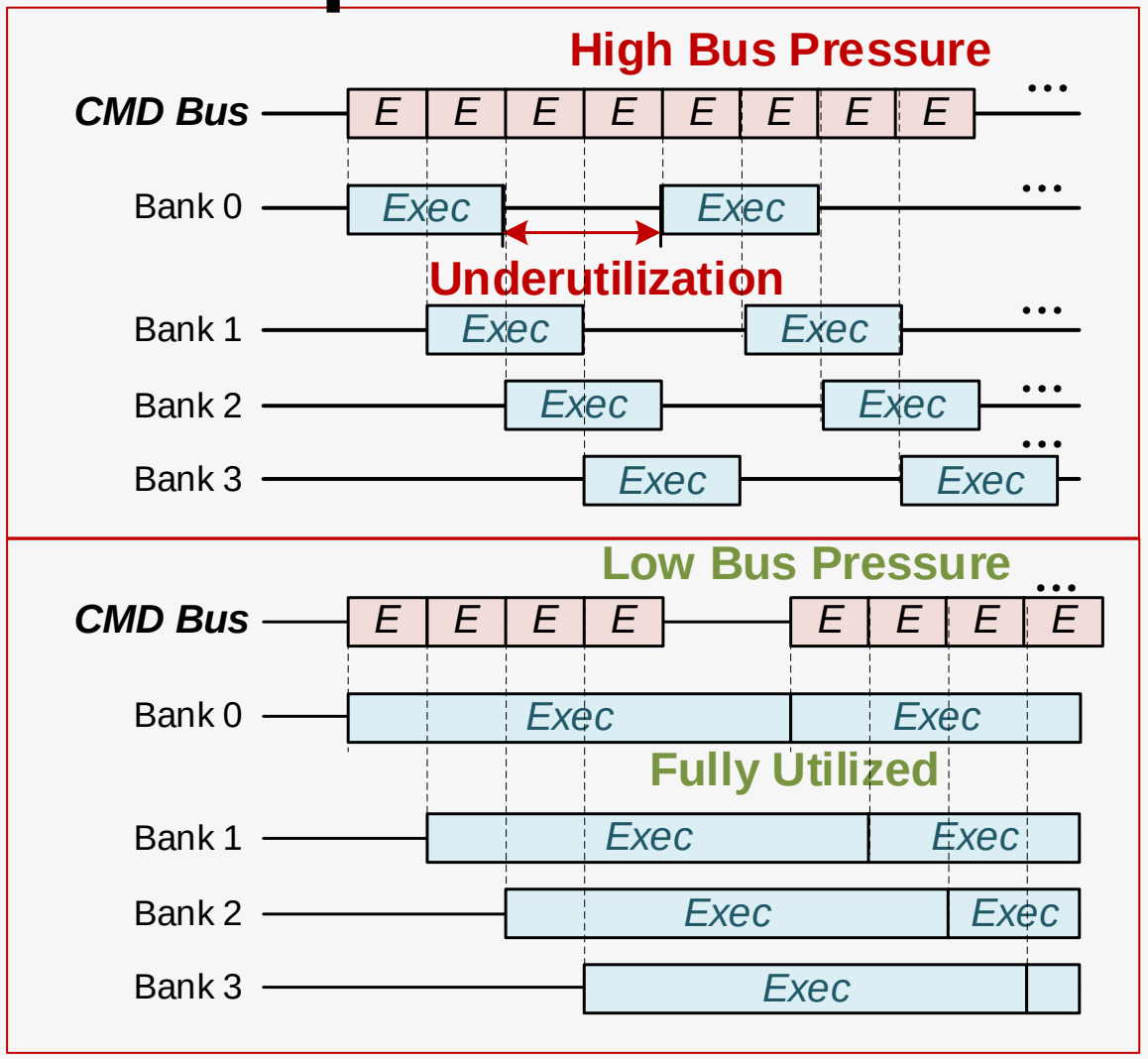


Observation 2

PIM underutilization due to bus pressure:

Short PIM execution command:

Long PIM execution command:



Challenge 1

Challenge 1: Dilemma of fixed-length commands

Command Length	CPU	PIM
Short	Low latency ✓	Bus Contention ✗
Long	High Latency ✗	High Throughput ✓

Interface Analysis II:

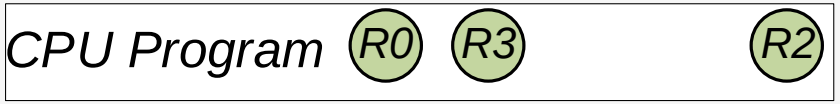
Interference between **PIM Data Transfer** & **CPU Access**

Observation 3

Co-schedule PIM data transfer and CPU access:

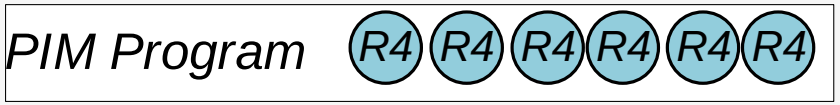
CPU Access v.s. PIM Transfer

Ri: Access to Row *i*



CPU access:

Random, Sporadic



PIM data transfer:

Contiguous, Bursty

Observation 3

Co-schedule PIM data transfer and CPU access:

FR-FCFS:
(First-ready, First-Come First-Serve)
A conventional memory scheduling policy

CPU Access v.s. PIM Transfer

Ri: Access to Row *i*

CPU Program R0 R3 R2

CPU access:

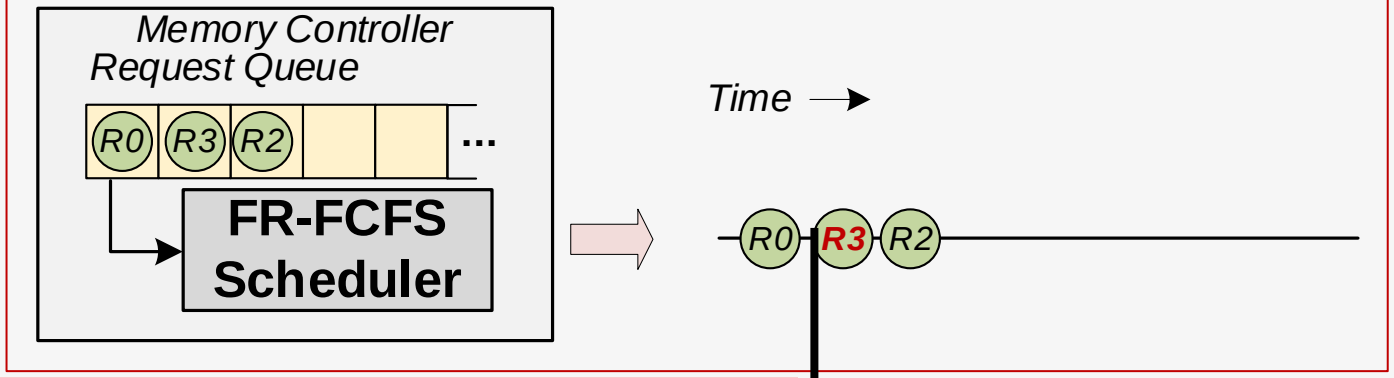
Random, Sporadic

PIM Program R4 R4 R4 R4 R4 R4

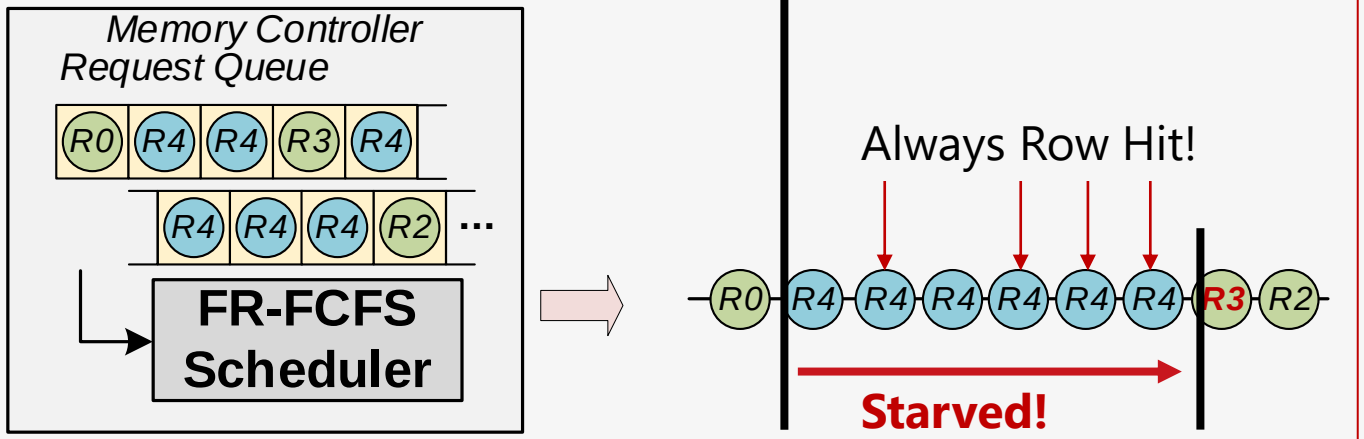
PIM data transfer:

Contiguous, Bursty

Only ***CPU***



Co-schedule ***CPU & PIM***



Observation 3

Co-schedule PIM data transfer and CPU access:

FR-FCFS:
(First-ready, First-Come First-Serve)
A conventional memory scheduling policy

CPU Access v.s. PIM Transfer

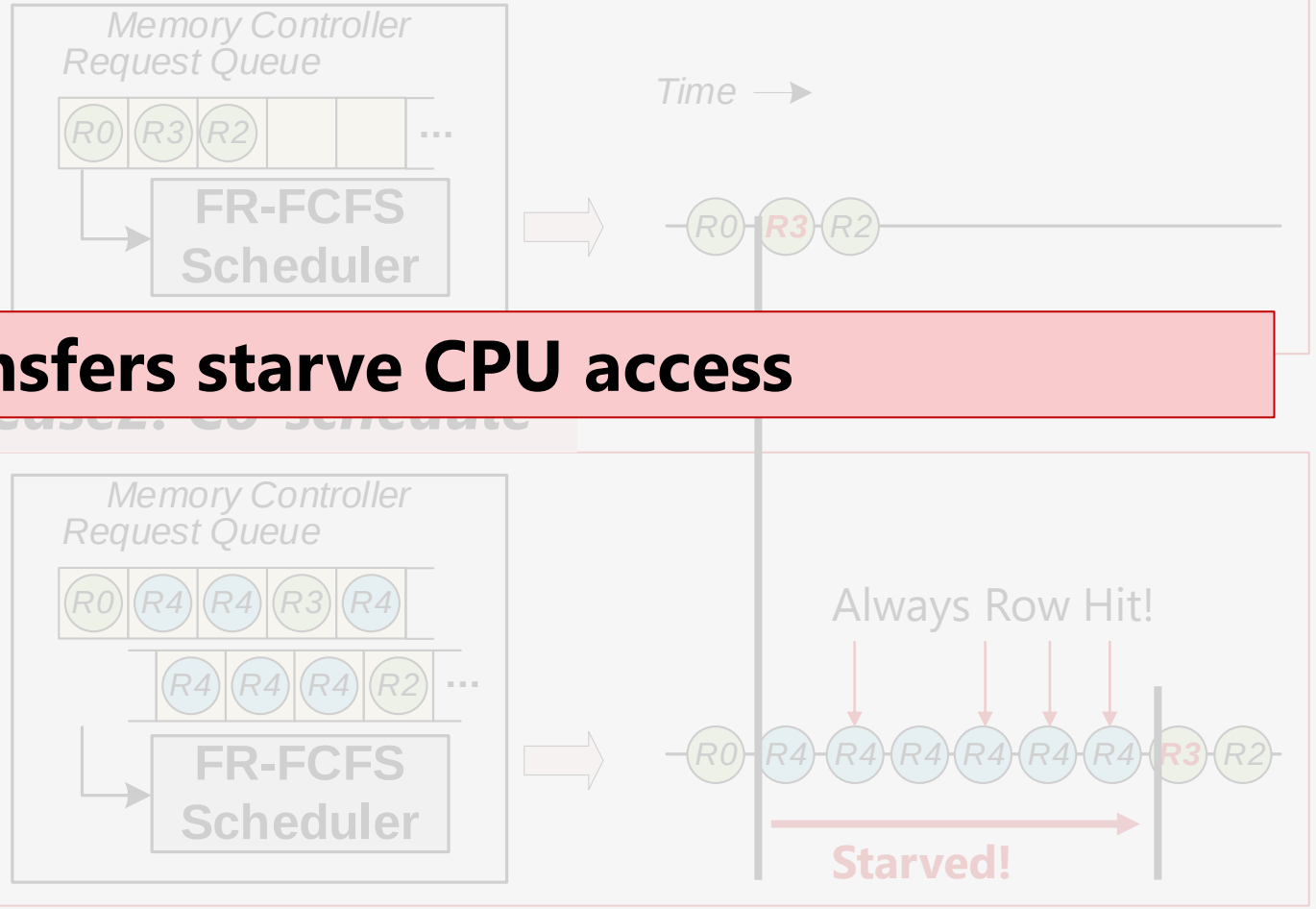
Ri: Access to Row *i*

CPU Program R0 R3 R2

PIM Program R4 R4 R4 R4 R4 R4

PIM data transfer:
Contiguous, Bursty

Case1: CPU Only



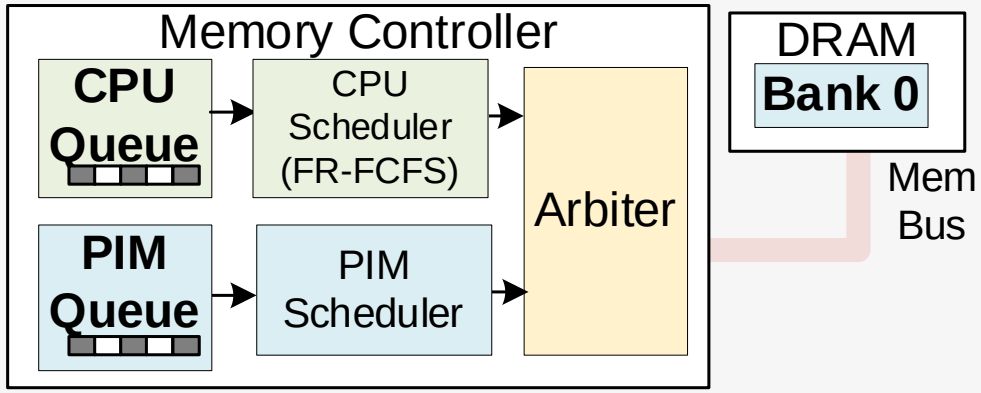
Challenge 2: PIM Data transfers starve CPU access

Scheduling Analysis III:
Unbalanced CPU Access & PIM Execution

Previous Work 1

Row-hit-aware Scheduling:

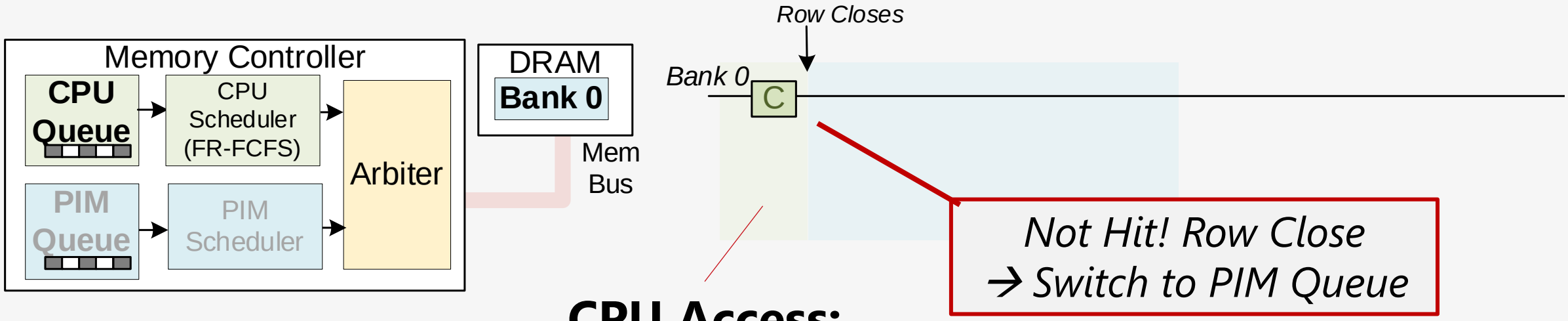
Switch between CPU/PIM queue **when row close**



Previous Work 1

Row-hit-aware Scheduling:

Switch between CPU/PIM queue **when row close**



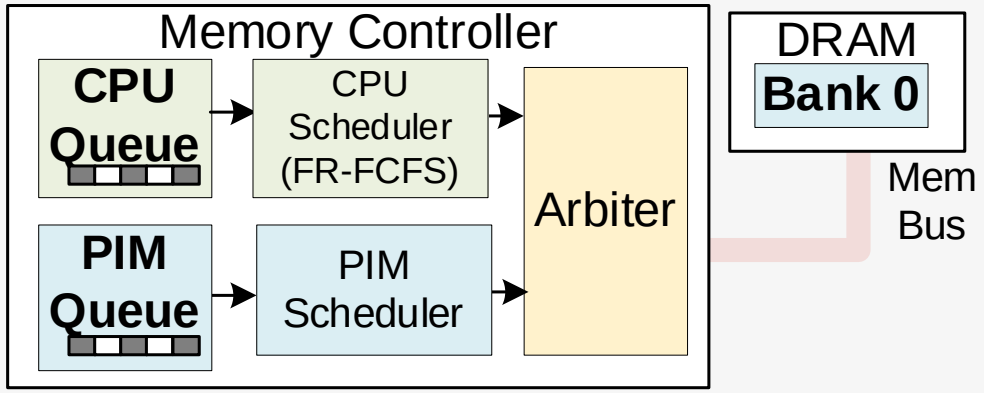
CPU Access:

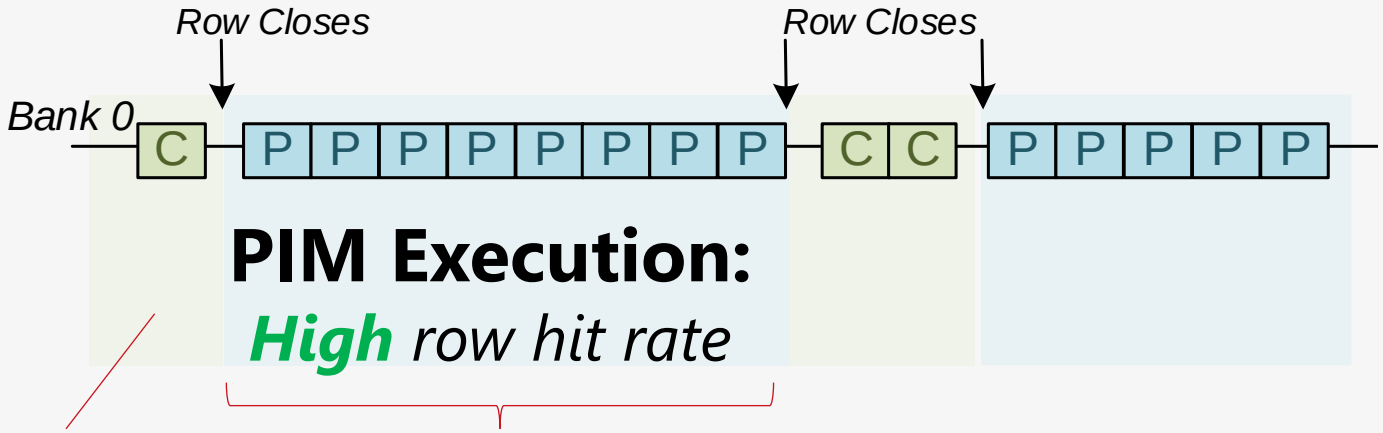
Low *row hit rate*

Previous Work 1

Row-hit-aware Scheduling:

Switch between CPU/PIM queue **when row close**





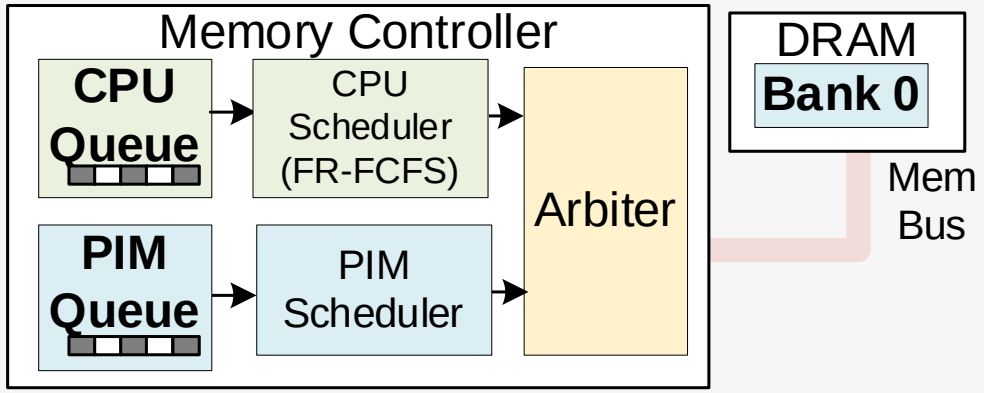
CPU Access:

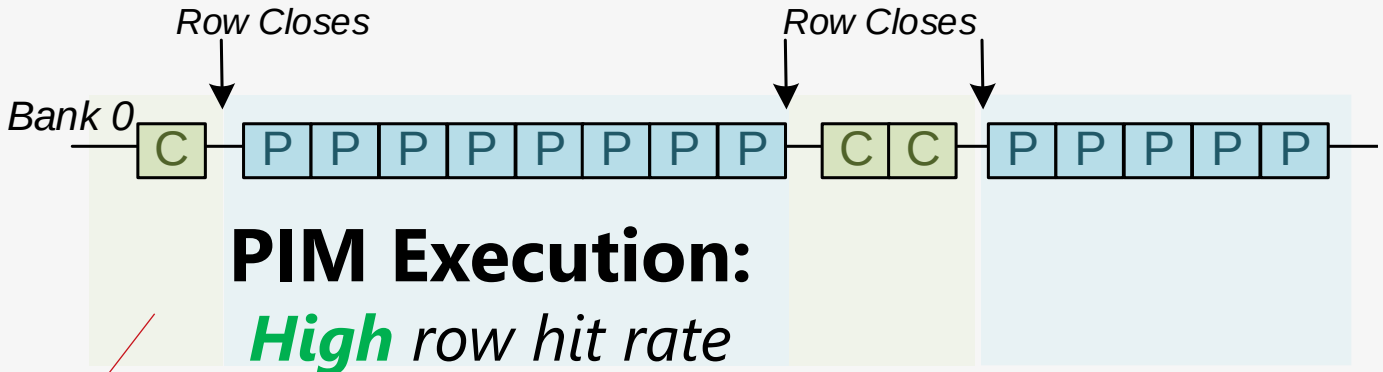
Low *row hit rate*

Previous Work 1

Row-hit-aware Scheduling:

Switch between CPU/PIM queue **when row close**





CPU Access:

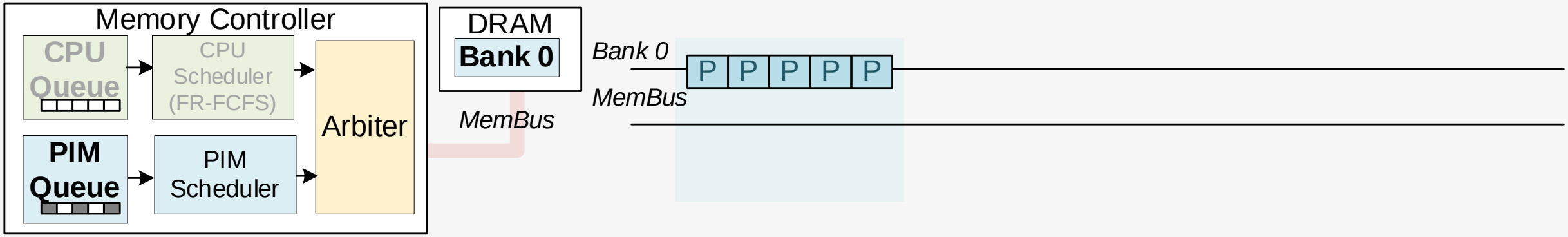
Low *row hit rate*

Biased to PIM Performance

Previous Work 2

CPU-first Scheduling:

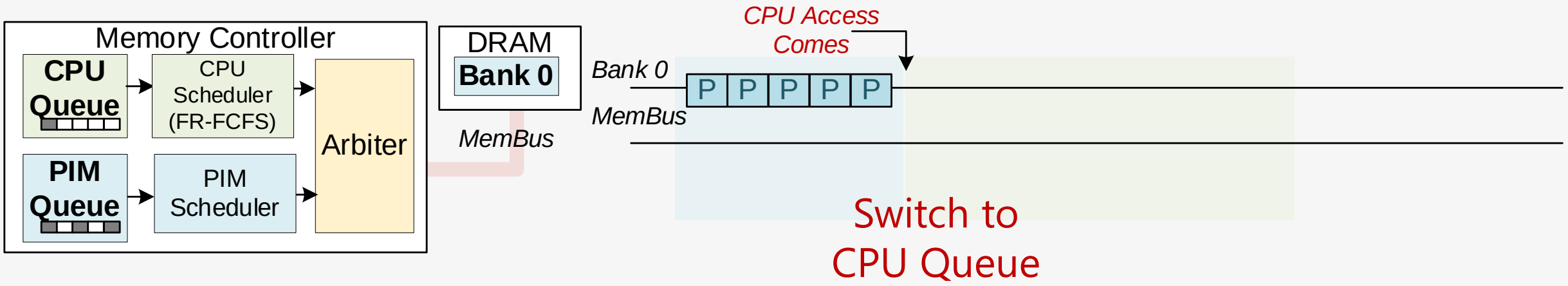
Only switch to PIM queue **when CPU queue is empty**



Previous Work 2

CPU-first Scheduling:

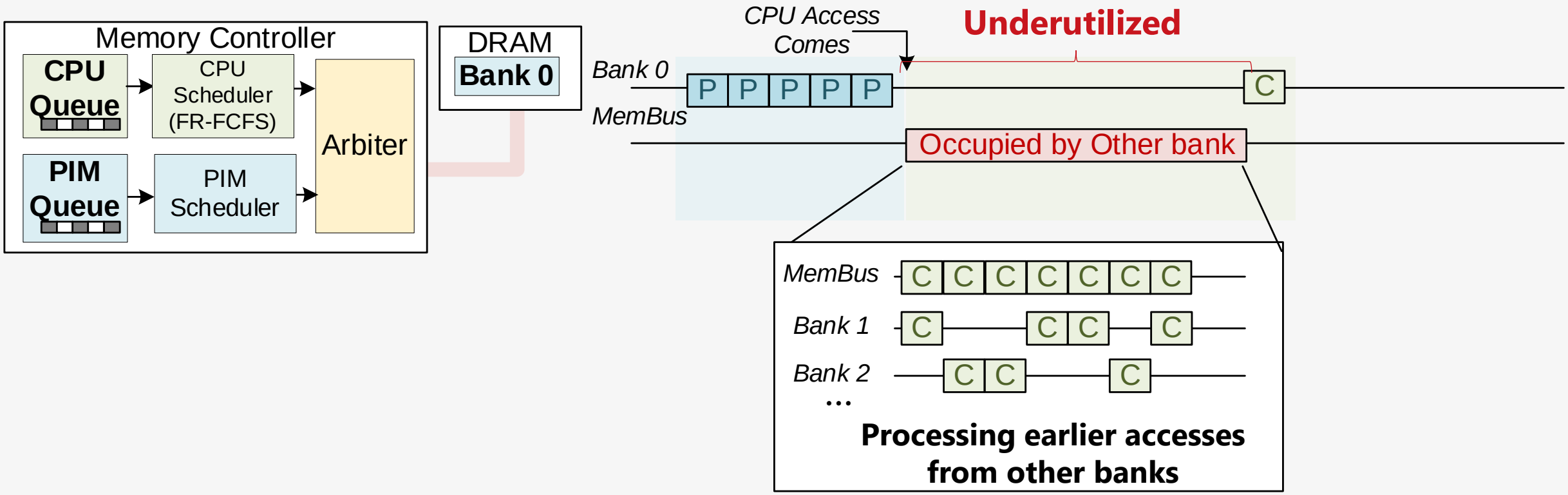
Only switch to PIM queue **when CPU queue is empty**



Previous Work 2

CPU-first Scheduling:

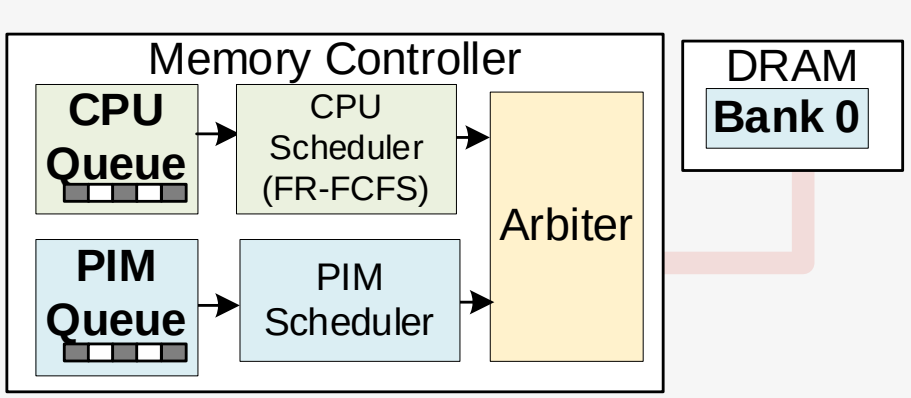
Only switch to PIM queue **when CPU queue is empty**

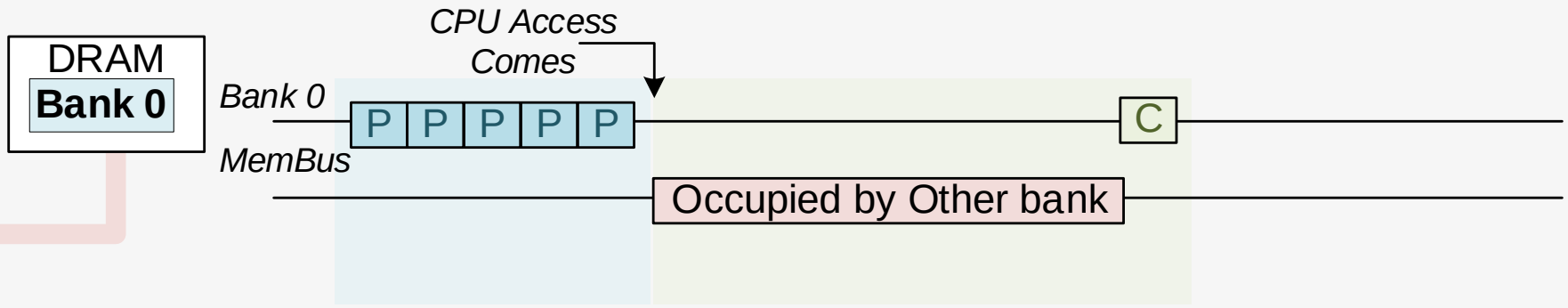


Previous Work 2

CPU-first Scheduling:

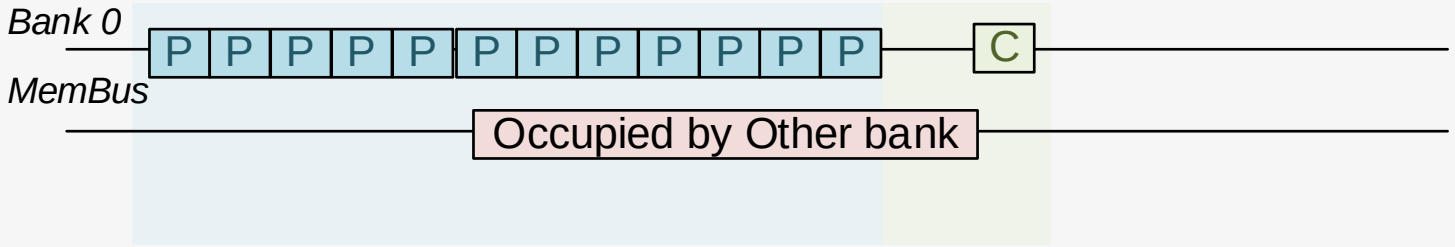
Only switch to PIM queue **when CPU queue is empty**





Delayed Switch is Better

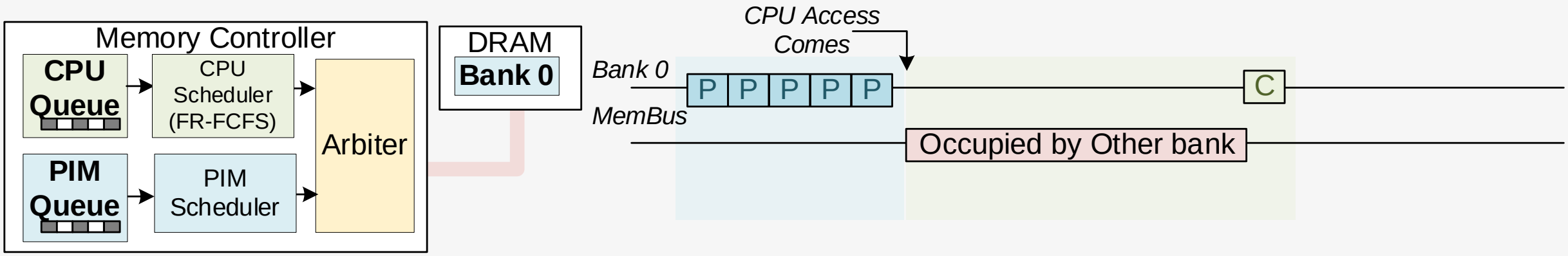
Issue more PIM executions



Previous Work 2

CPU-first Scheduling:

Only switch to PIM queue **when CPU queue is empty**



Biased to CPU Performance

Challenge 3

Row-hit-aware:
Biased to **PIM**

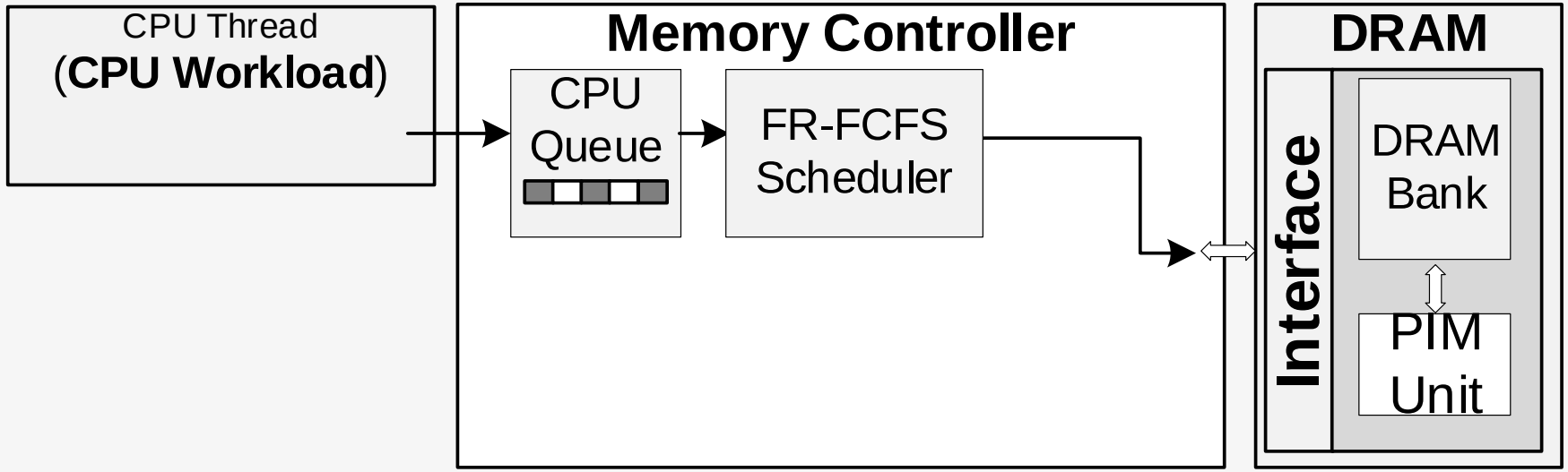
CPU-first:
Biased to **CPU**

Challenge 3: Simultaneously achieve high CPU & PIM performance

COSM: A Cooperative Scheduling Framework

Co-optimize Interface & Scheduling

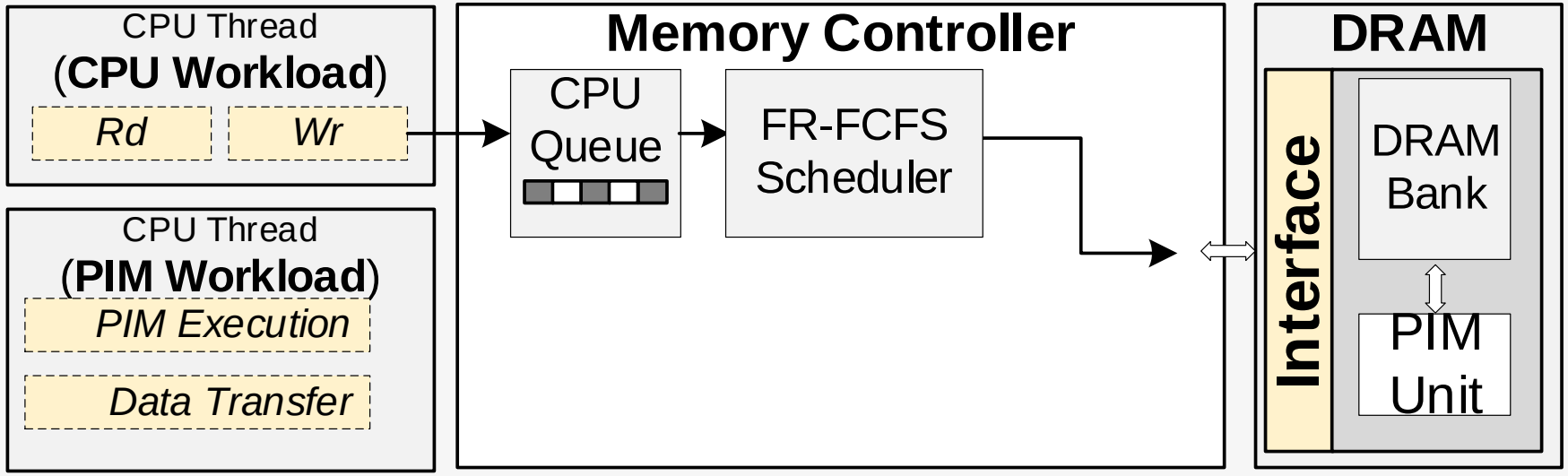
Challenge ***Solution***



COSM: A Cooperative Scheduling Framework

Co-optimize Interface & Scheduling

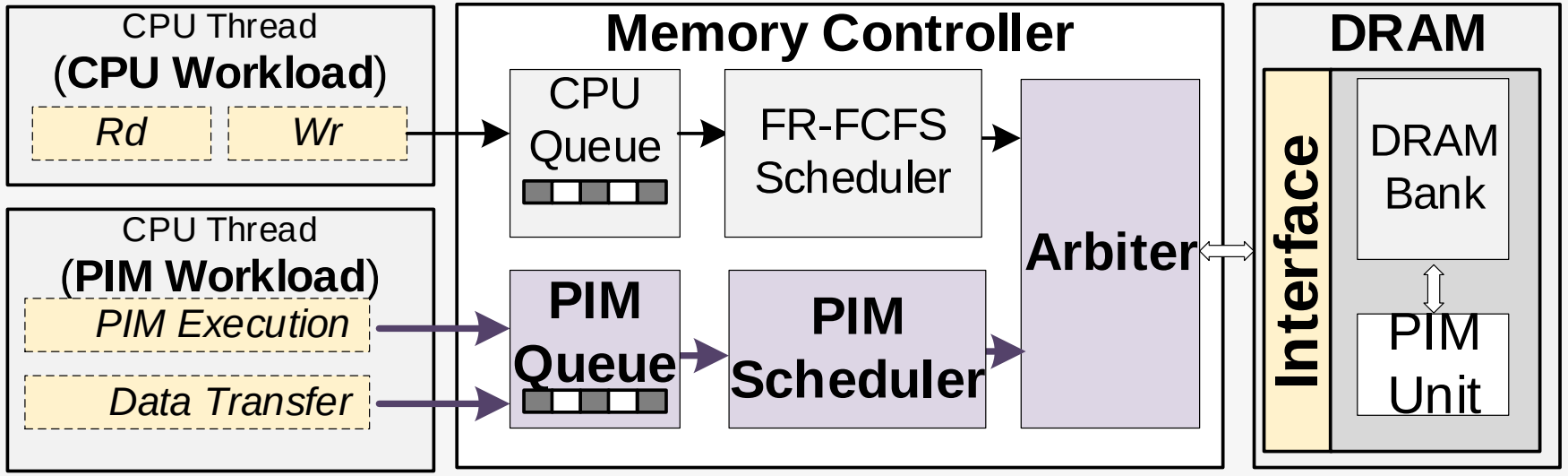
Challenge		*Solution*
		Low-interference *Interface*
1 Dilemma of fixed-length commands	→	Preemptable PIM Execution
2 PIM data transfers starve CPU access	→	Bandwidth-decoupled Data Transfer



COSM: A Cooperative Scheduling Framework

Co-optimize Interface & Scheduling

Challenge		*Solution*
		Low-interference *Interface*
1 Dilemma of fixed-length commands	→	Preemptable PIM Execution
2 PIM data transfers starve CPU access	→	Bandwidth-decoupled Data Transfer
3 Balance CPU & PIM performance	→	**Idleness-aware Memory *Scheduling***

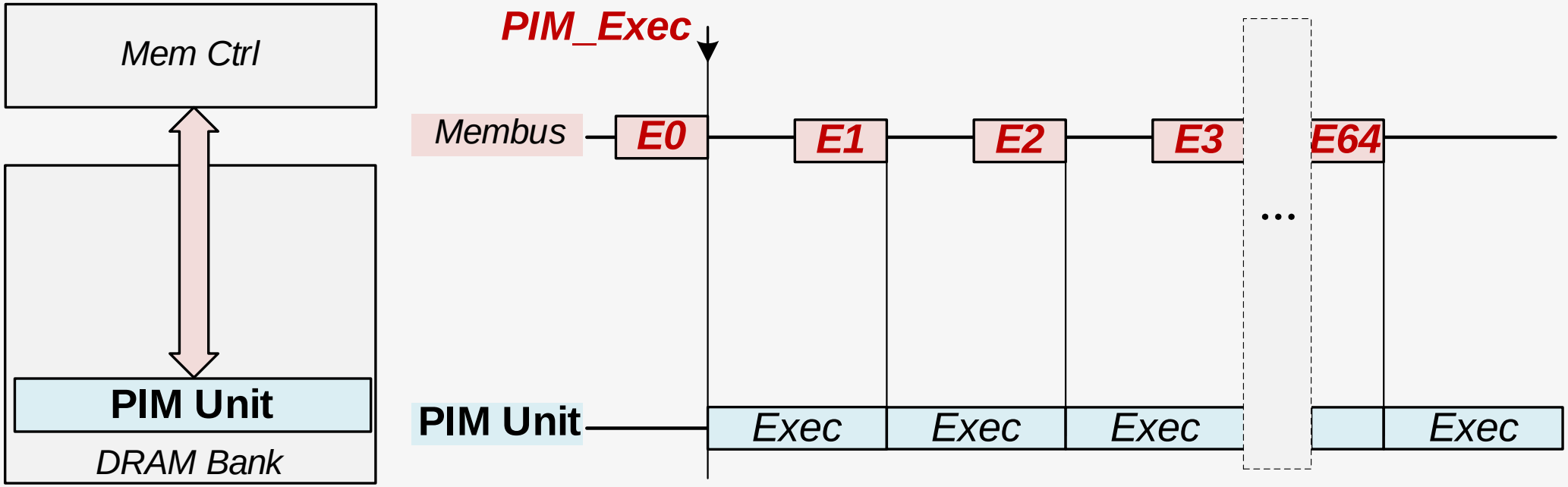


Interface— 1.1 Preemptable PIM Execution (1)

***Solution** of Challenge 1*: **Fixed-length → Preemptable**

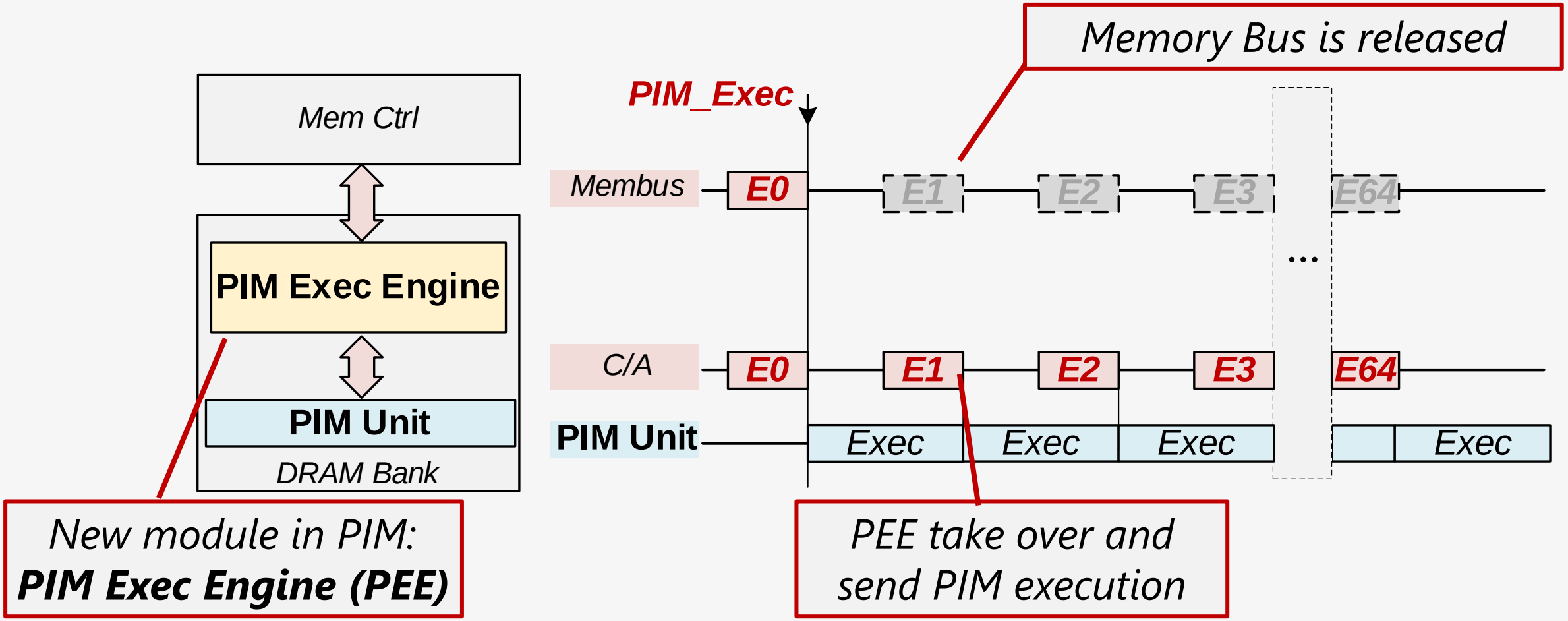
Interface— 1.1 Preemptable PIM Execution (1)

***Solution** of Challenge 1*: **Fixed-length → Preemptable**



Interface— 1.1 Preemptable PIM Execution (1)

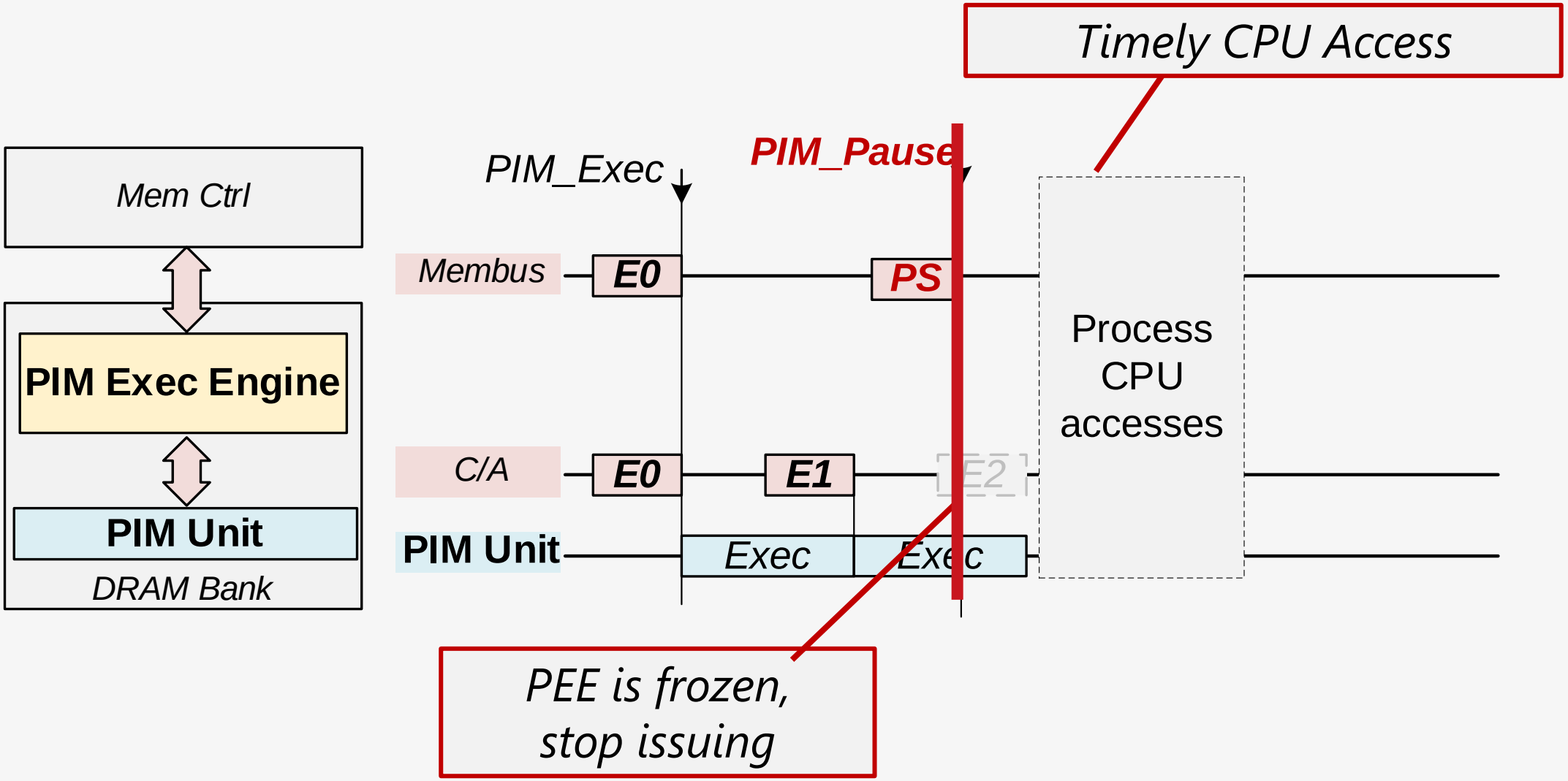
***Solution of Challenge 1*: Fixed-length → Preemptable**



Interface— 1.1 Preemptable PIM Execution

CPU access preempts PIM execution

1. Memory controller send a ***PIM_Pause*** to **freeze** PEE



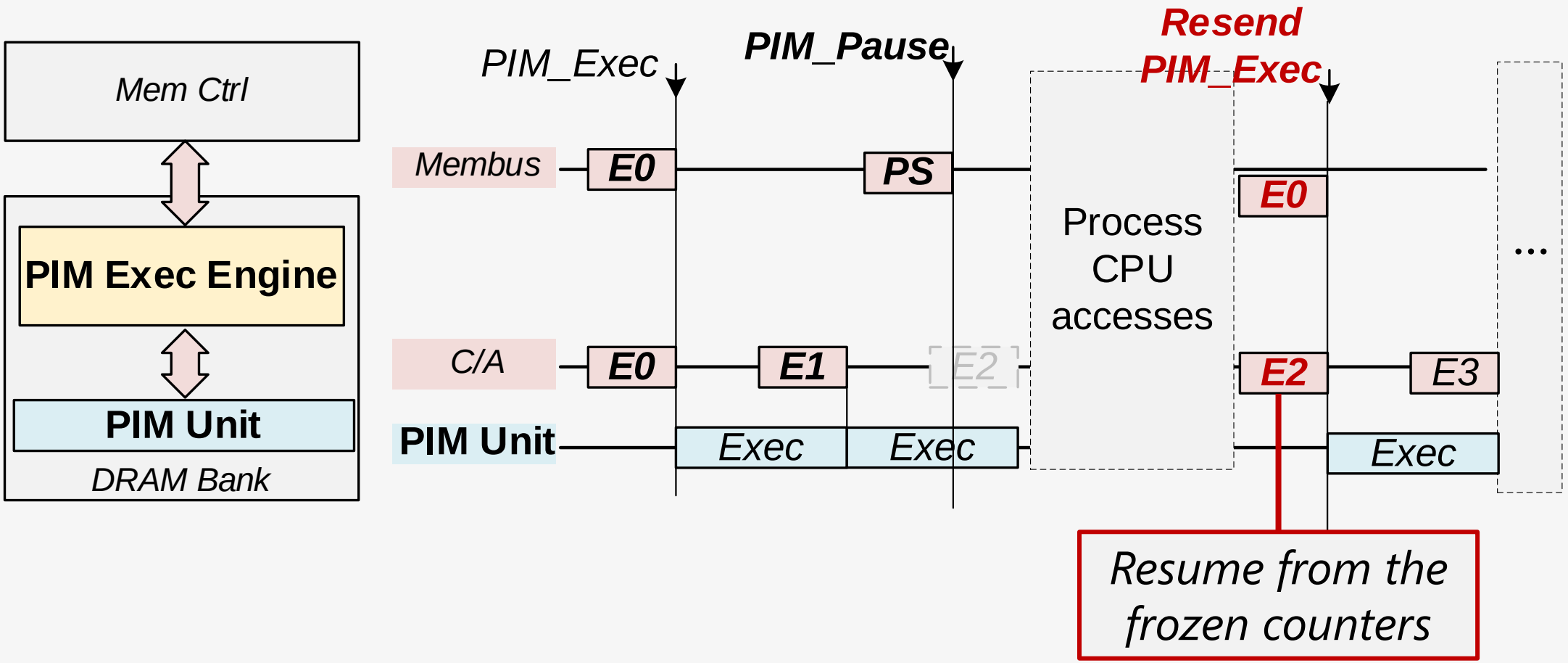
Interface— 1.1 Preemptable PIM Execution

CPU access preempts PIM execution

1. Memory controller send a ***PIM_Pause*** to **freeze** PEE

After CPU accesses are done:

2. Memory controller resend ***PIM_Exec*** to **resume** PEE



Interface— 1.2 Bandwidth-decoupled Data Transfer (1)

Should prioritize CPU access over PIM data transfer to avoid CPU access starving

Interface— 1.2 Bandwidth-decoupled Data Transfer (1)

Should prioritize CPU access over PIM data transfer to avoid CPU access starving

But using normal access command for PIM data transfer is **NOT efficient.**

Because it requires **strict** timing synchronization:

1 The **external** bus must be **idle**

2 **AND** the target row of **internal** bank must be **pre-opened**

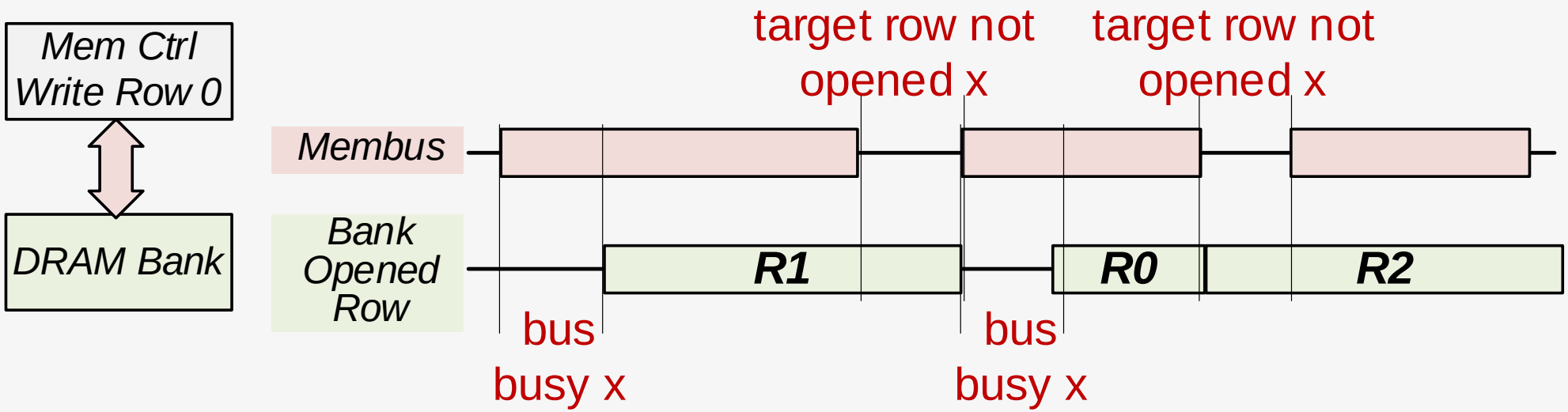
→ *Missing **any** window leads to transfer failure.*

Interface— 1.2 Bandwidth-decoupled Data Transfer (1)

Should prioritize CPU access over PIM data transfer to avoid CPU access starving

But using normal access command for PIM data transfer is **NOT efficient.**

Because it requires **strict** timing synchronization:
1 The **external** bus must be **idle**
2 **AND** the target row of **internal** bank must be **pre-opened**
→ *Missing **any** window leads to transfer failure.*

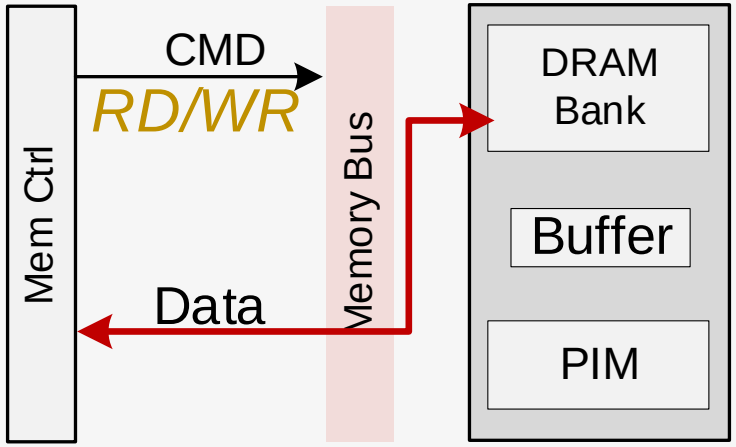


Numerous idle windows exist, but they are scarcely concurrent.

Interface— 1.2 Bandwidth-decoupled Data Transfer (2)

***Solution*: Decouple internal (bank) & external (bus) bandwidth occupation**

Conventional
(through normal access)



Occupy both external & internal

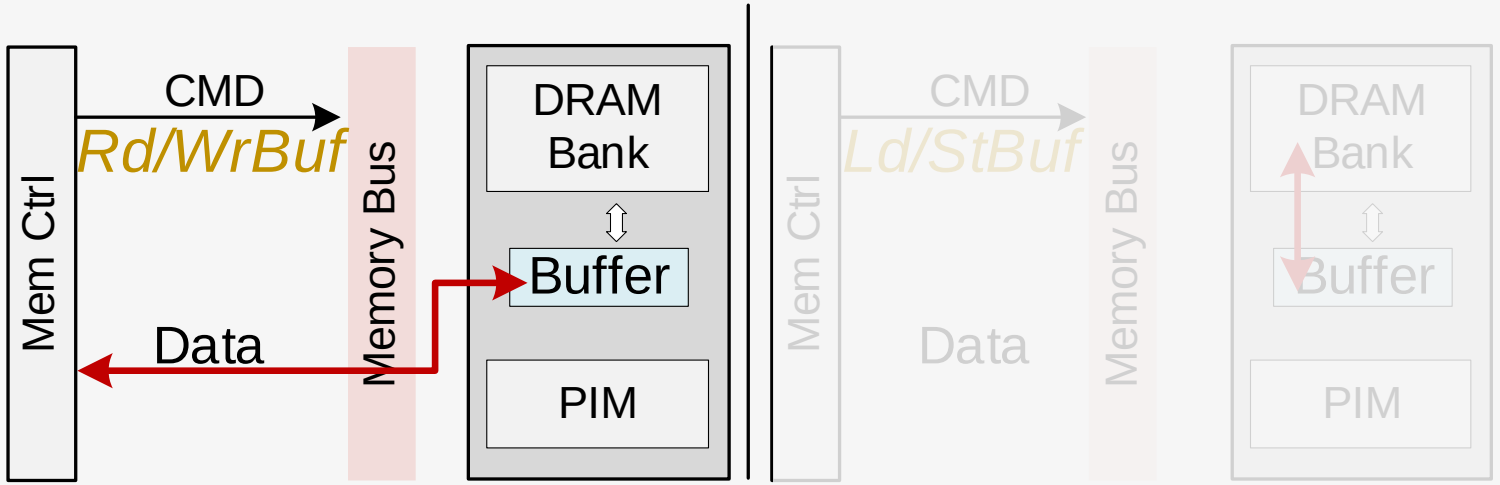
Interface— 1.2 Bandwidth-decoupled Data Transfer (2)

***Solution*: Decouple internal (bank) & external (bus) bandwidth occupation**

Conventional
(through normal access)

Occupy both external & internal

Bandwidth-decoupled Data Transfer



***External* only**
Host <->PIM Buffer

***Internal* only**
PIM Buffer<-> Bank

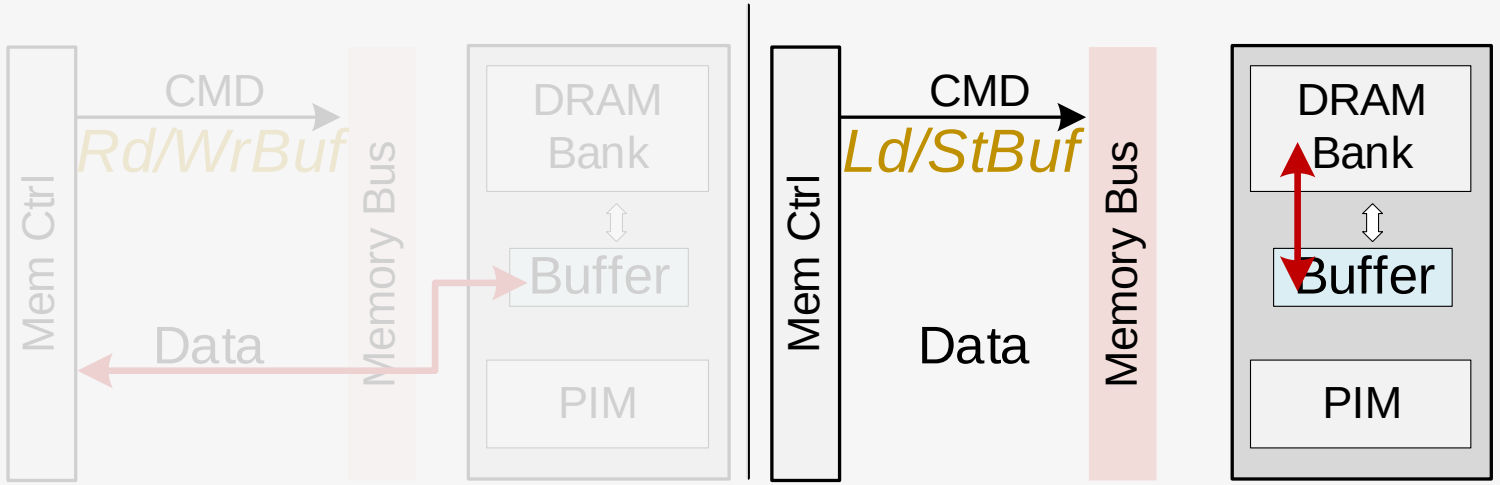
Interface— 1.2 Bandwidth-decoupled Data Transfer (2)

***Solution*: Decouple internal (bank) & external (bus) bandwidth occupation**

Conventional
(through normal access)

Occupy both external & internal

Bandwidth-decoupled Data Transfer



External only
Host <->PIM Buffer

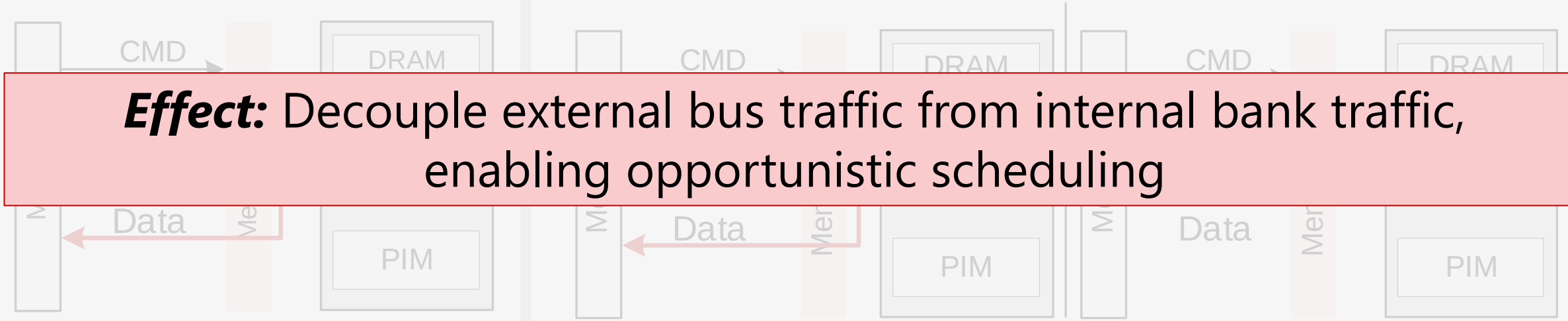
Internal only
PIM Buffer<-> Bank

Interface— 1.2 Bandwidth-decoupled Data Transfer (2)

***Solution*: Decouple internal (bank) & external (bus) bandwidth occupation**

Conventional
(through normal access)

Bandwidth-decoupled Data Transfer



Effect: Decouple external bus traffic from internal bank traffic, enabling opportunistic scheduling

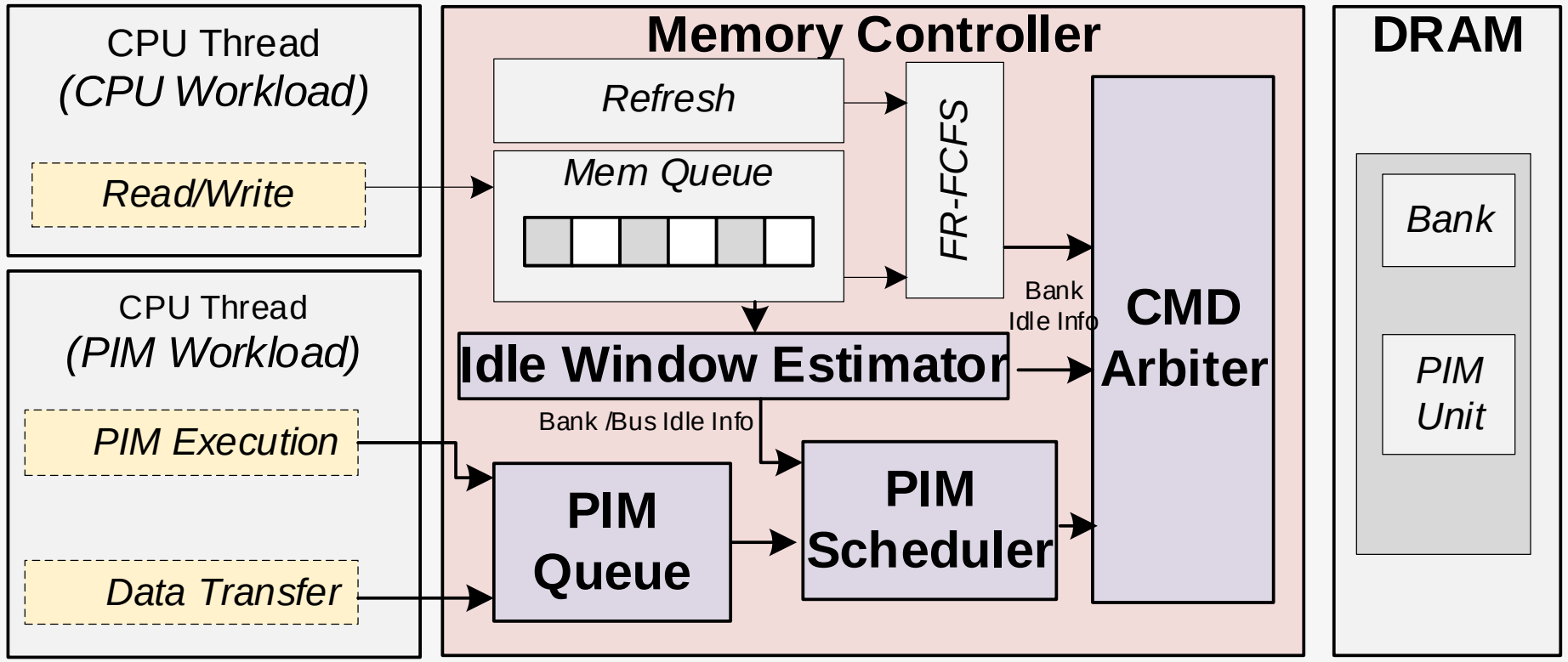
Occupy both external & internal

External only
Host <->PIM Buffer

Internal only
PIM Buffer<-> Bank

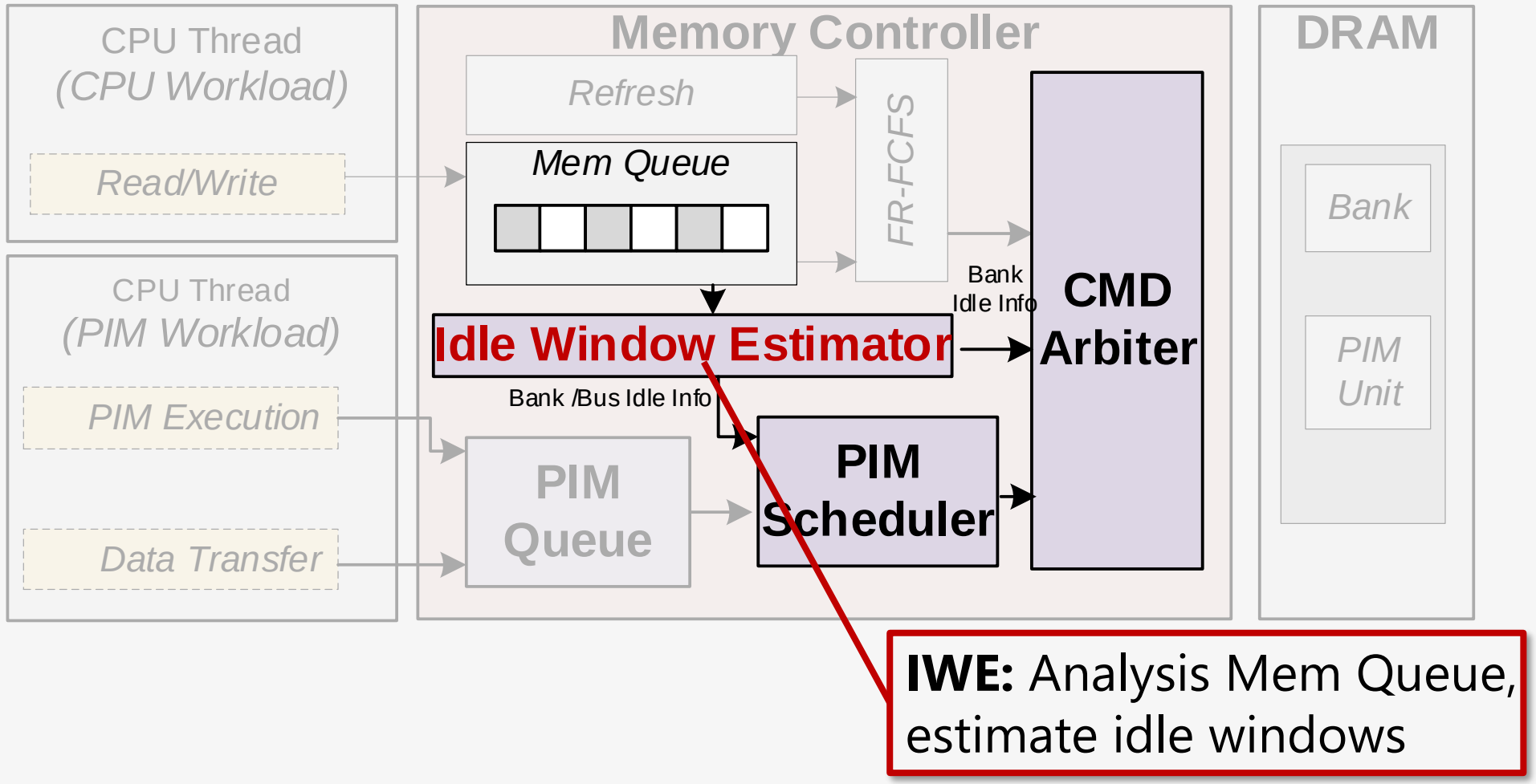
2 Idleness-aware Memory Scheduling (1)

Locate CPU access idle windows, insert PIM command into the windows



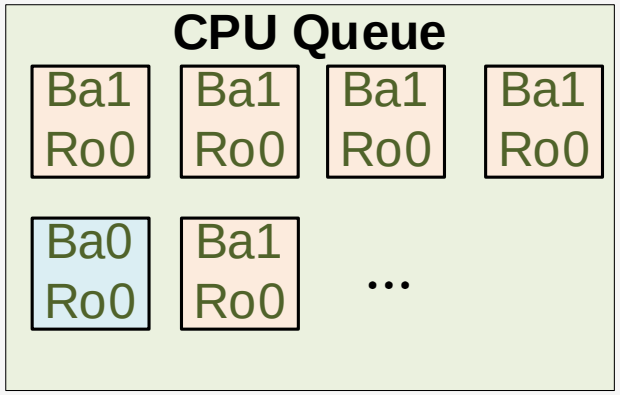
2 Idleness-aware Memory Scheduling (1)

Locate CPU access idle windows, insert PIM command into the windows

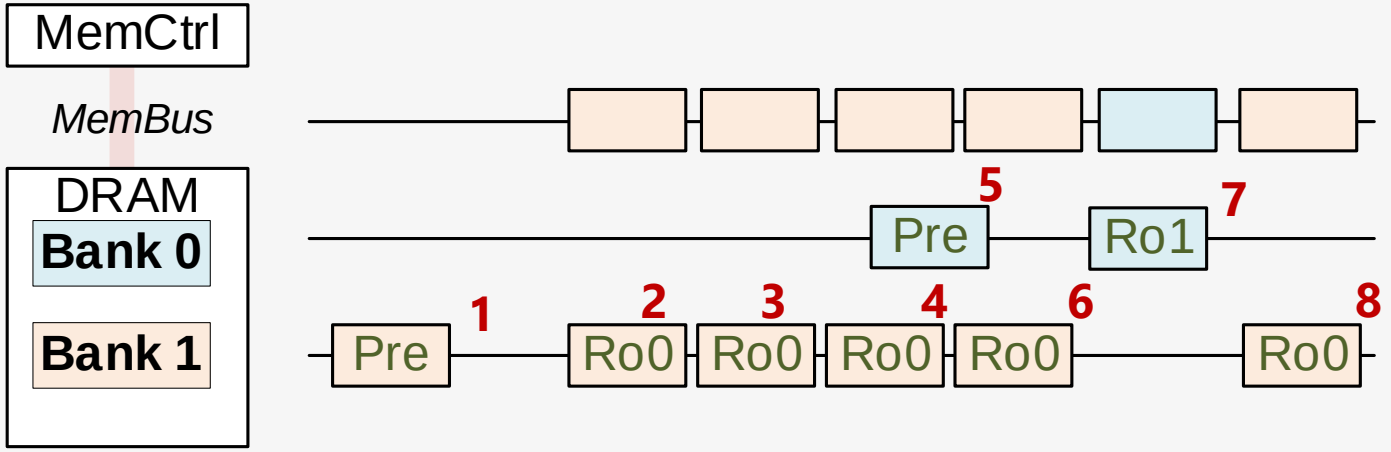


Idle Window Estimator (IWE)

Analysis CPU Queue

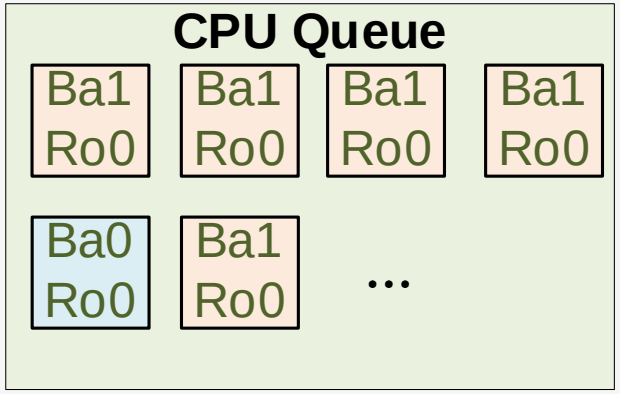


Step 1: Predict Issue Order
(According to CPU Scheduler, e.g., FR-FCFS)

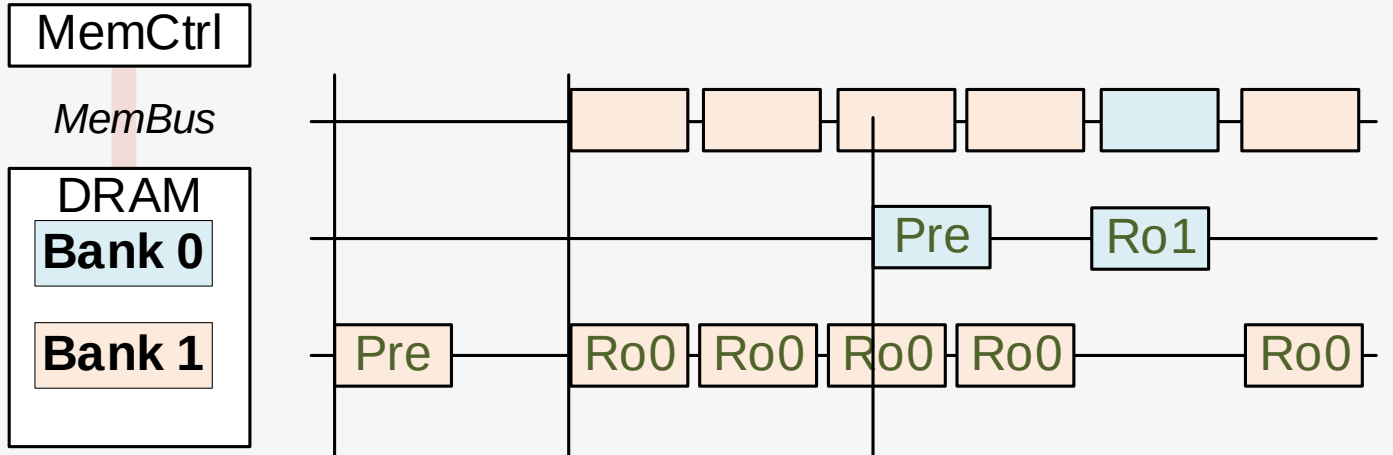


Idle Window Estimator (IWE)

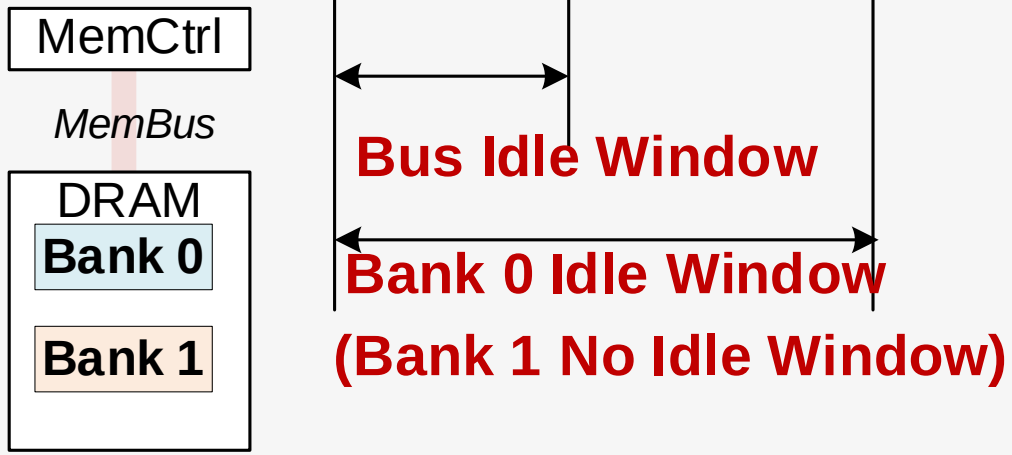
Analysis CPU Queue



Step 1: Predict Issue Order
(According to CPU Scheduler, e.g., FR-FCFS)

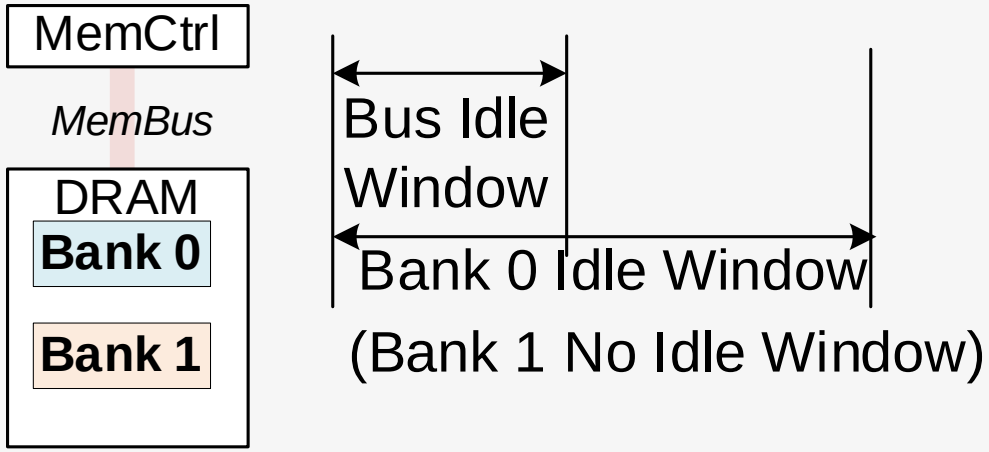


Step 2: Estimate Idle Time Window



Idle Window Estimator (IWE)

Step 2: Estimate Idle Time Window



Use these information

If the window is **larger than a threshold,** scheduler will
Insert PIM commands **exactly to the idle time window** (**Avoid bias to PIM**)
Delay CPU Pre-charge and send more PIM Execution (**Avoid bias to CPU**)

More in the Paper

- Extended analysis of related work (All-bank PIM interface)
- **In-depth details on idleness-aware scheduling**, including:
  - Algorithm for estimating idle time windows
  - Overlapped execution phase to better utilize idle time windows
- Discussion on the implications for the software stack

Experimental Setup

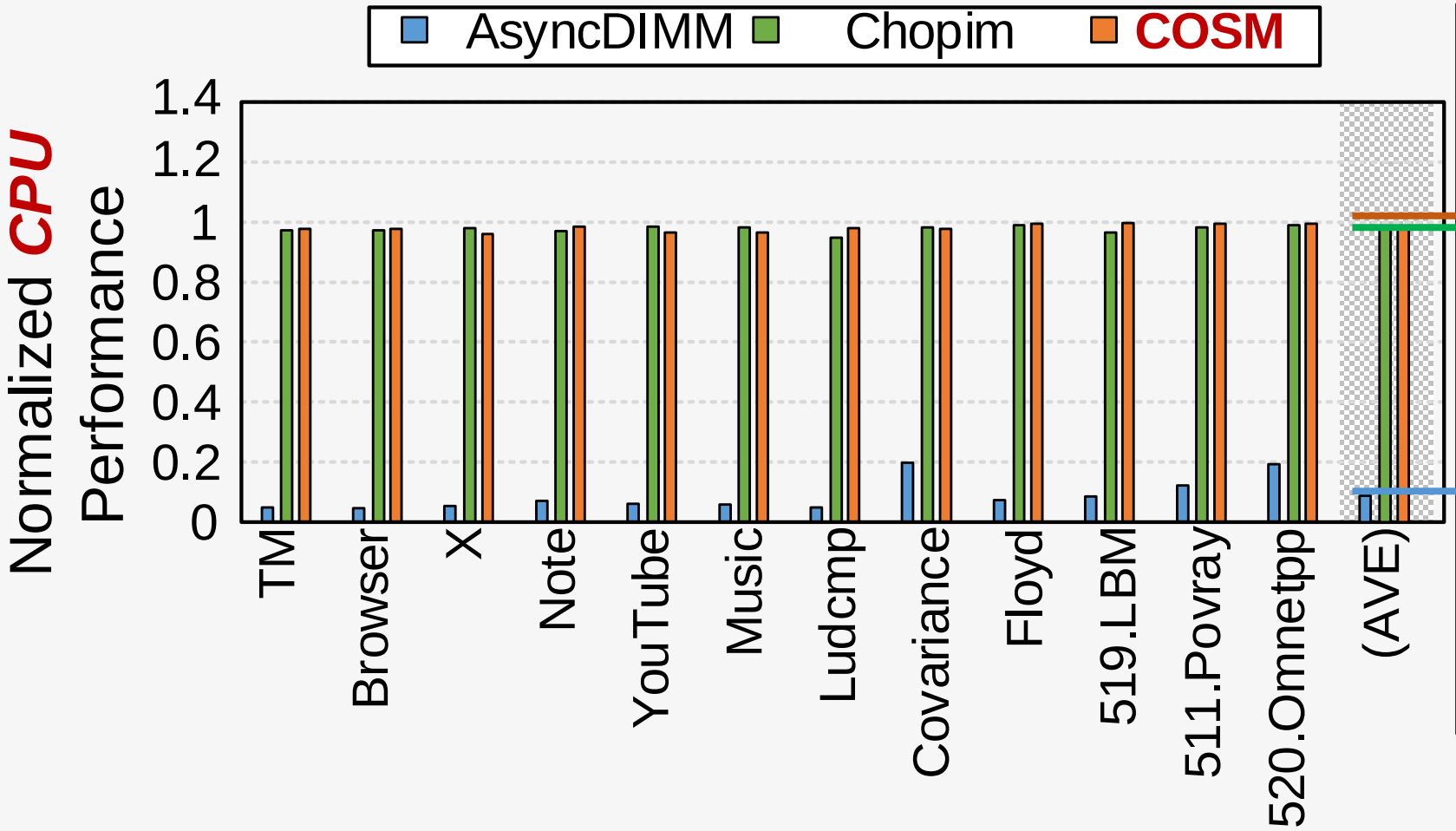
- ***Benchmarks:***
  PIM Benchmarks: BLOOM-1B1, DeepSeek-R1-1.5B, and Qwen2-0.5B.
  CPU Benchmarks: 6 mobile APPs, 3 PolyBench (contiguous access), 3 SPEC 2017
- ***Baselines:***
  ASYNCDIMM [HPCA'25]:row-hit-aware scheduling
  Chopim [ISCA'20]: CPU-first scheduling

- ***Simulation:***
  CPU model: collect trace on a mobile phone
  DRAM model: Ramulator2
  Proposed Scheduling Modules: Synopsys DC

- ***System Configuration:***

System Configuration	
CPU	8 cores @2.1Hz on average (Follow Xiaomi 11 Pro)
DRAM	LPDDR5, 2 Channel*2 Ranks
PIM	32GOPS, 64 per Rank (Follow Samsung LPDDR5-PIM)

Result: CPU Performance



CPU Degradation:

COSM: 2.0%

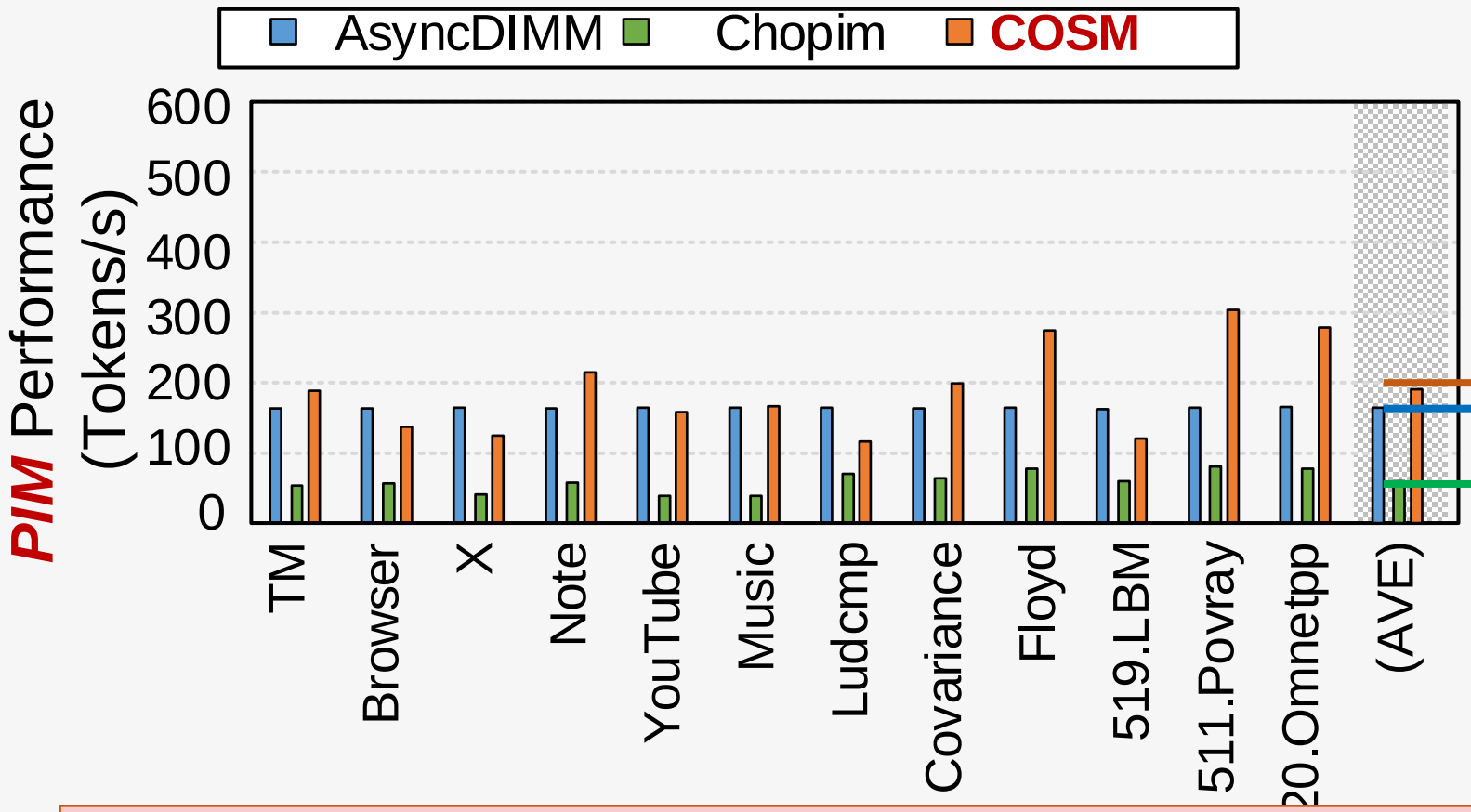
Chopim(CPU-first): <3%

AsyncDIMM
(Row-hit-aware): > 90%

LLM: BLOOM 1b1

More results in paper: Results on other LLM models

Result: PIM Performance



PIM Performance:

COSM
1.4x over **AsyncDIMM**
(Row-hit-aware)
2.8x over **Chopim**
(CPU-first)

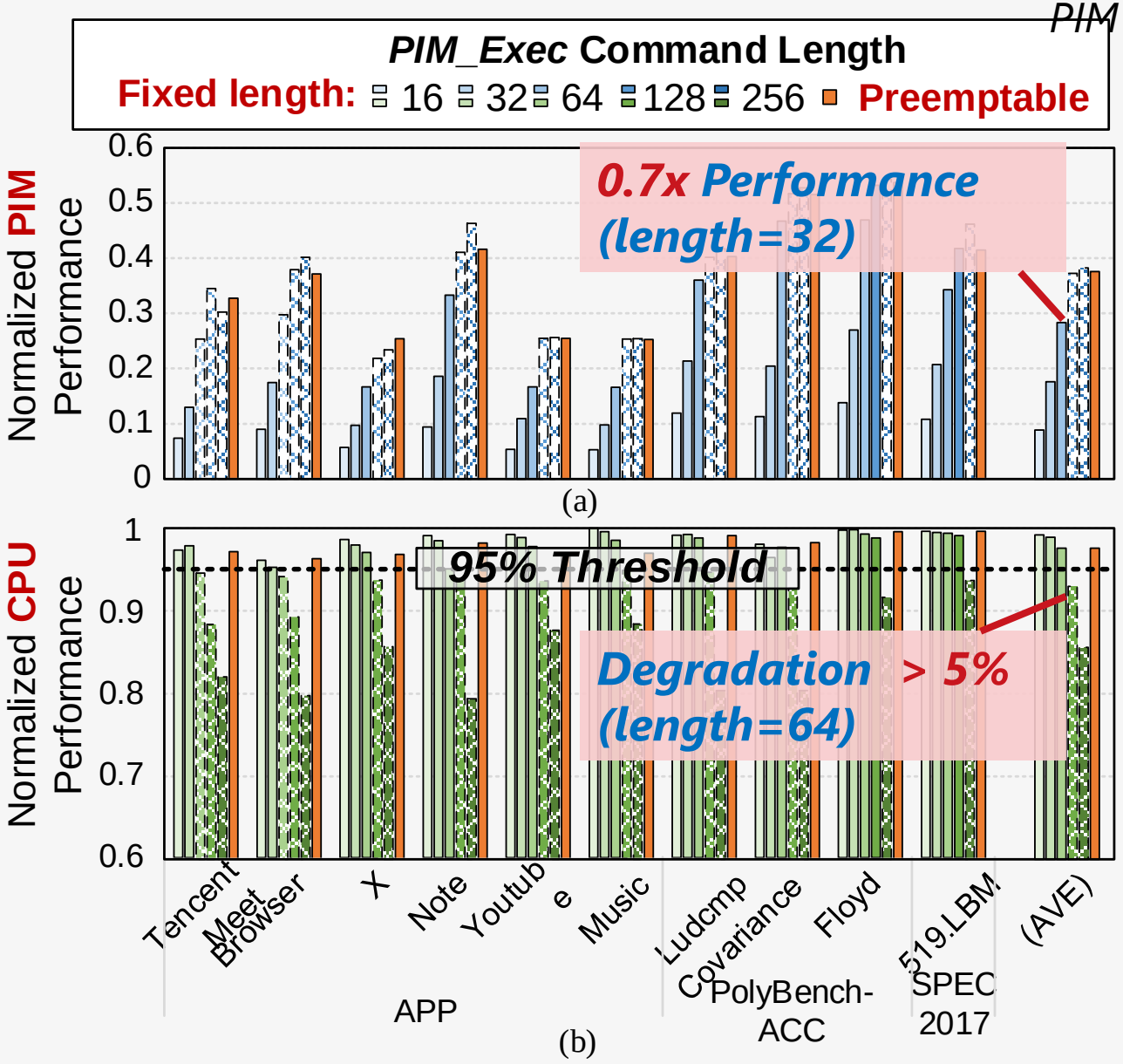
COSM achieves 2.8x PIM Throughput with only 2% CPU Degradation.

LLM: BLOOM-7B1

More results in paper: Results on other LLM models

Result: The Effect of Preemptable Execution

PIM workload: KQV gen of DeepSeek (Exclude data transfer)



Fixed-length commands:

Length = 32:

0.7 x *Max PIM Performance*

CPU Degradation ***< 5%***

Length = 64:

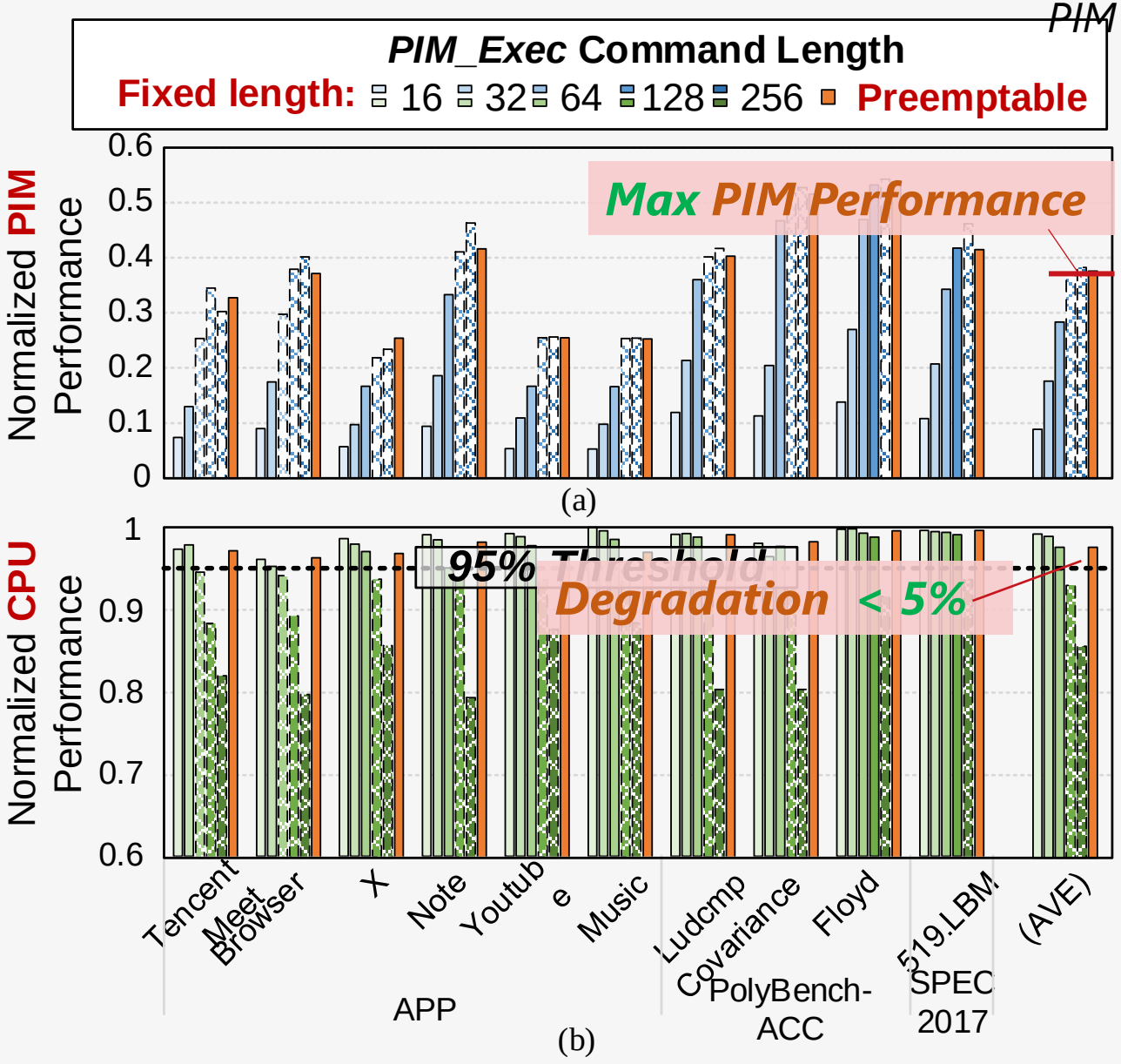
Max *PIM Performance*

CPU Degradation ***> 5%***

-- Trade-off

Result : The Effect of Preemptable Execution

PIM workload: KQV gen of DeepSeek (Exclude data transfer)



Preemptable commands:

Max *PIM Performance*

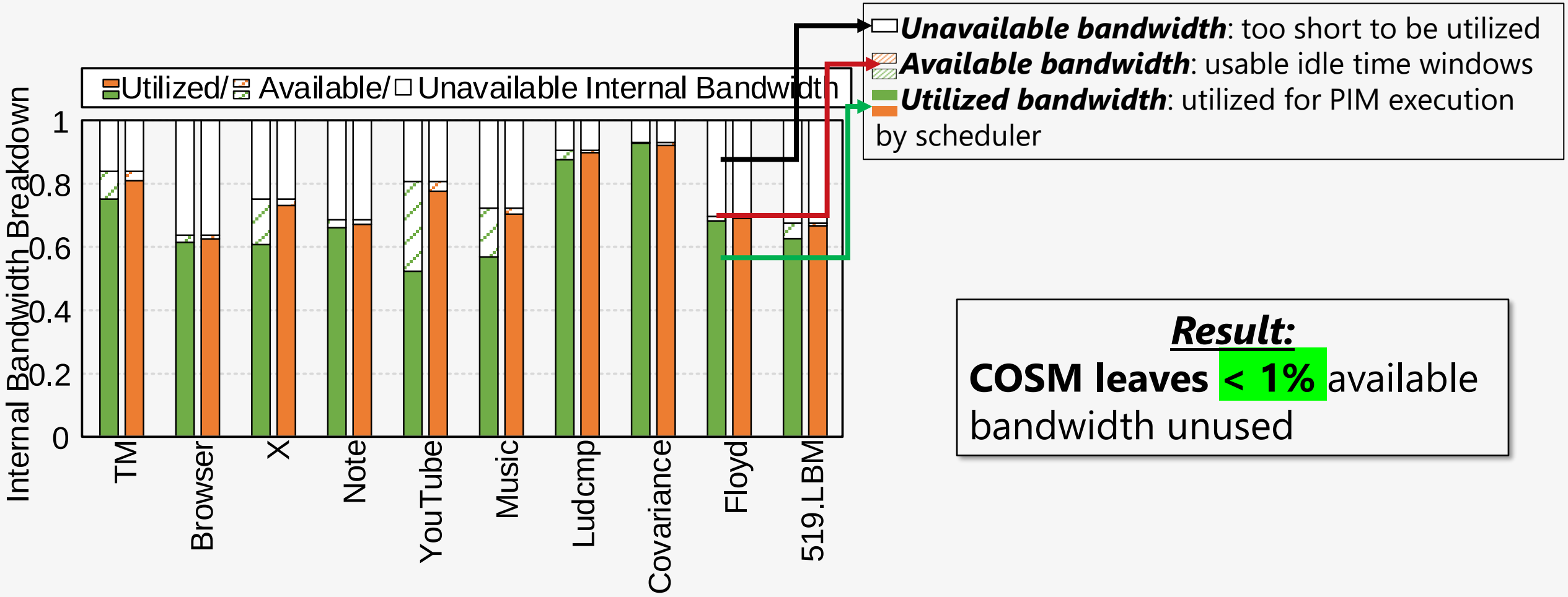
CPU Degradation ***< 5%***

Breaks the CPU-PIM trade-off

More results in paper: Results on other LLM models

Result: Effect of Idleness-aware Scheduling

PIM workload: Attention of DeepSeek, ***only PIM Execution***



Result:

COSM leaves < 1% available bandwidth unused

COSM can better utilize the idle time windows

Area

Synthesized with Synopsys Design Compiler; scaled to 5nm

Components	Area	Overhead
Original Memory Controller	**~0.93mm²**	-
PIM Scheduler	0.014 mm^2	
Idle Time Estimator	0.0085 mm^2	
Command Arbiter	0.0054 mm^2	
PIM Queues	0.041 mm^2	
Total	**0.069 mm^2**	**7.4%**

Conclusion

A Cooperative **Scheduling Framework** for Concurrent PIM & CPU Execution on Mobile Devices.

Critical contributions:

- Low-interference **Interface**
- Idleness-aware Memory **Scheduling**

Results: 2.8x PIM Throughput with only 2% CPU Degradation.

Artifacts Available

Artifacts Evaluated - Functional

Results Reproduced

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Thank You!

饮 水 思 源　爱 国 荣 校

COSM: A Cooperative Scheduling Framework for Concurrent PIM and CPU Execution on Mobile Devices

Yilong Zhao(Speaker)*, **Fangxin Liu***, **Onur Mutlu, Mingyu Gao, Jian Liu, Haibing Guan, Li Jiang†**

sjtuzyl@sjtu.edu.cn , liufangxin@sjtu.edu.cn, ljiang_cs@sjtu.edu.cn

Shanghai Jia Tong University | Shanghai Qi Zhi Institute
ETH Zurich | Tsinghua University | Beihang University

ISCA' 2026 July 6, 2026

* *Co-first Author*
† Corresponding Author