PCMCsim: An Accurate Phase-Change Memory Controller Simulator and its Performance Analysis

ISPASS 2022

Hyokeun Lee*, Hyungsuk Kim*, Seokbo Shim‡, Seungyong Lee*,
Dosun Hong‡, Hyuk-Jae Lee*, Hyun Kim₱

* Seoul National University
₱ Seoul National University of Science and Technology
‡ SK Hynix

Outline

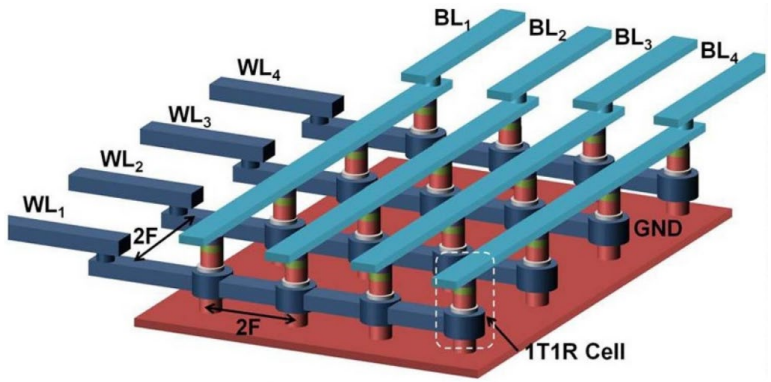
- Background & Goal
- Motivation
- Our simulator: PCMCsim
  - System overview
  - Execution flow
  - Optimization: AIT Manager Prefetcher
- Evaluation
  - Accuracy validation
  - Takeaway messages
- Conclusion

Outline

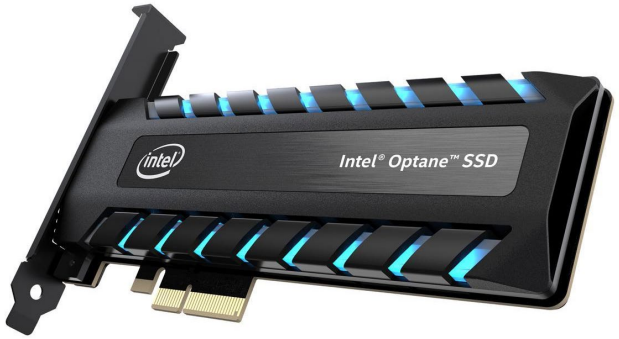
- **Background & Goal**
- Motivation
- Our simulator: PCMCsim
  - System overview
  - Execution flow
  - Optimization: AIT Manager Prefetcher
- Evaluation
  - Accuracy validation
  - Takeaway messages
- Conclusion

Background

- Phase-change memory (PCM or PRAM)
  - Non-volatile + High density
    - Material phase → Resistance value
    - “1T1R” → No capacitor
  - Promising next-generation memory/storage



- Use of PCM
  - Fast storage
    - Mount DAXFS
    - Product: Intel Optane DC SSD
  - Low-cost main memory pool expansion
    - Further reduce cost → disaggregated memory
    - Product: Intel Optane DC PMM



Simulation of PCM in previous studies

- Modifications on processor simulators (e.g., McSimA+)
  - Modify memory latency values
  - **Few timing parameters → low simulation accuracy**
- Modifications of DRAM simulators (e.g., DRAMsim)
  - More timing parameters, modified in detail
  - **Slow (cycle-level) + No features for PCM controller**
- Dedicated simulator (e.g., NVMain)
  - Simulation of cell-level characteristics
  - **vs. modern PCM module? → Old structure…**

Our goals

- **Develop a PCM controller simulator achieving following three goals**

- Goal1: Incorporating **all necessary features** of modern PCM modules
  - (1) Hardware path; (2) address indirection table (AIT); (3) read-modify-write (RMW)
  - Different from other memory controller architectures

- Goal2: **Fast and accurate** PCM controller simulator
  - Event-driven is generally faster than cycle-level
  - Accuracy validation against real hardware is required

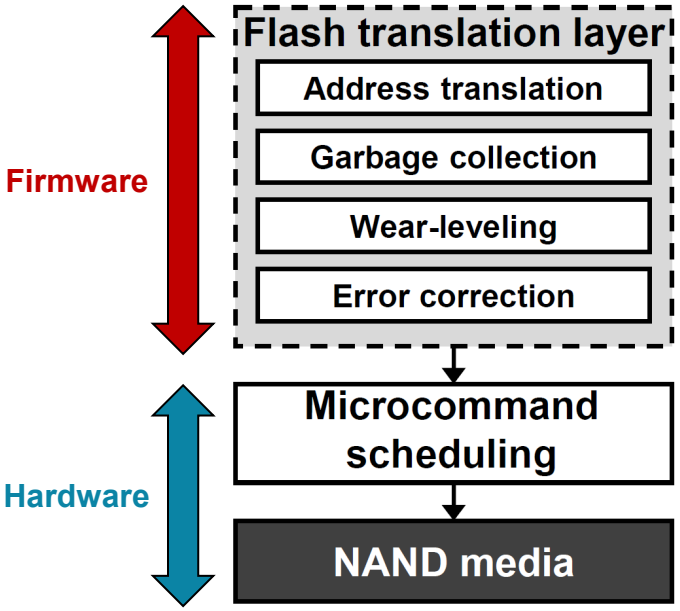
- Goal3: **Performance characterization** of new simulator
  - New architecture → new performance characteristics
  - Providing key takeaway messages + Optimization

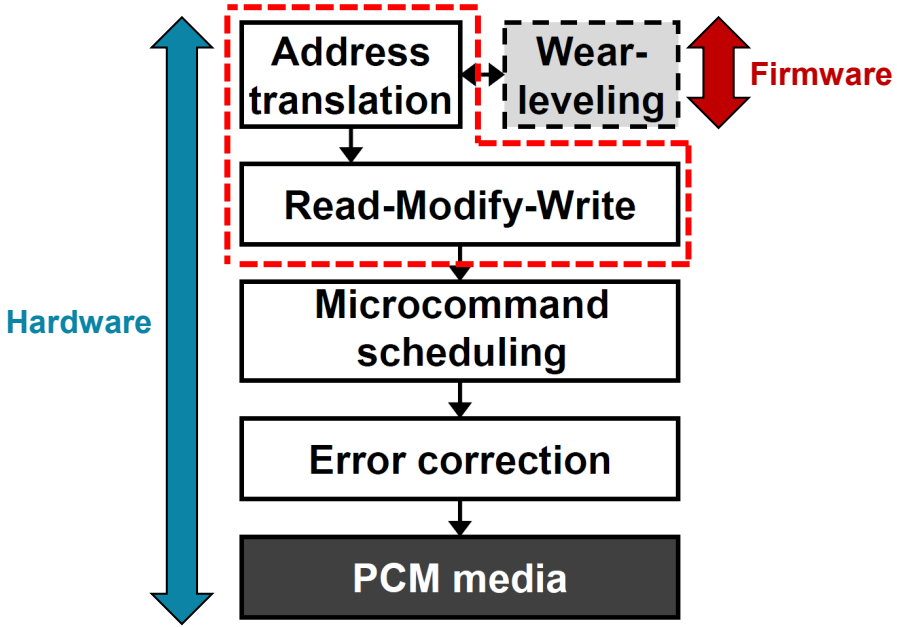
Outline

- Background & Goal
- **Motivation**
- Our simulator: PCMCsim
  - System overview
  - Execution flow
  - Optimization: AIT Manager Prefetcher
- Evaluation
  - Accuracy validation
  - Takeaway messages
- Conclusion

Necessity of PCM controller features

- **Three features are required** for a modern PCM controller
  - AIT: mapping table that translates host address to device address
  - RMW: manage gap between host cacheline size and PCM access granularity
  - Hardware path: process host requests mainly with hardware pipeline
- Reasons for incorporating three features
  - NAND mainly processes with firmware → **wrong performance characteristics**
  - AIT and RMW account for large overhead → **new research directions**





Reducing latency of accessing AIT

- Accessing AIT incurs **high latency overhead**

- An **address translation hardware** needs to access DRAM-subsystem
  - More than 20% of total latency
  - Considerable in PCM-based systems
    - Naively thinking: **DRAM latency = 44.6 ns** vs. **PCM latency = 150 ns**
    - Practical: go through not only DRAM but also interconnect
  - Also, our evaluation result (later) shows similar results

- A low-cost speedup scheme for AIT access is required

Outline

- Background & Goal
- Motivation
- **Our simulator: PCMCsim**
  - System overview
  - Execution flow
  - Optimization: AIT Manager Prefetcher
- Evaluation
  - Accuracy validation
  - Takeaway messages
- Conclusion

Our simulator: PCMCsim

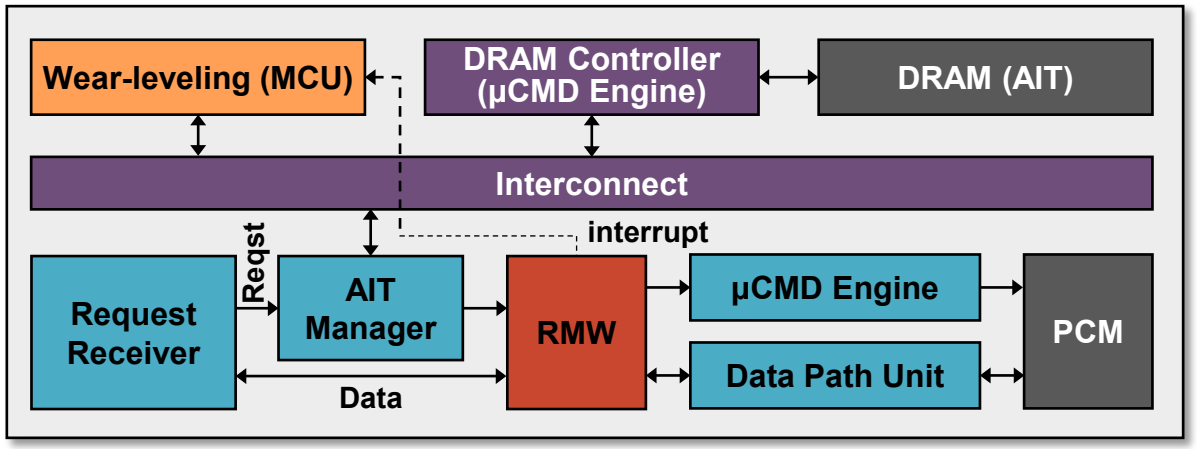
- PCMCsim?
  - Simulates features for modern PCM controller architecture
  - Event-driven simulator
  - Functional- and cycle-accurate simulator
- Modular implementation for scalability
  - Instantiation of a DRAM subsystem for AIT
  - Capable of simulating DRAM in a standalone manner
- Facilitates design exploration of various capabilities of PCM

Outline

- Background & Goal
- Motivation
- **Our simulator: PCMCsim**
  - **System overview**
  - Execution flow
  - Optimization: AIT Manager Prefetcher
- Evaluation
  - Accuracy validation
  - Takeaway messages
- Conclusion

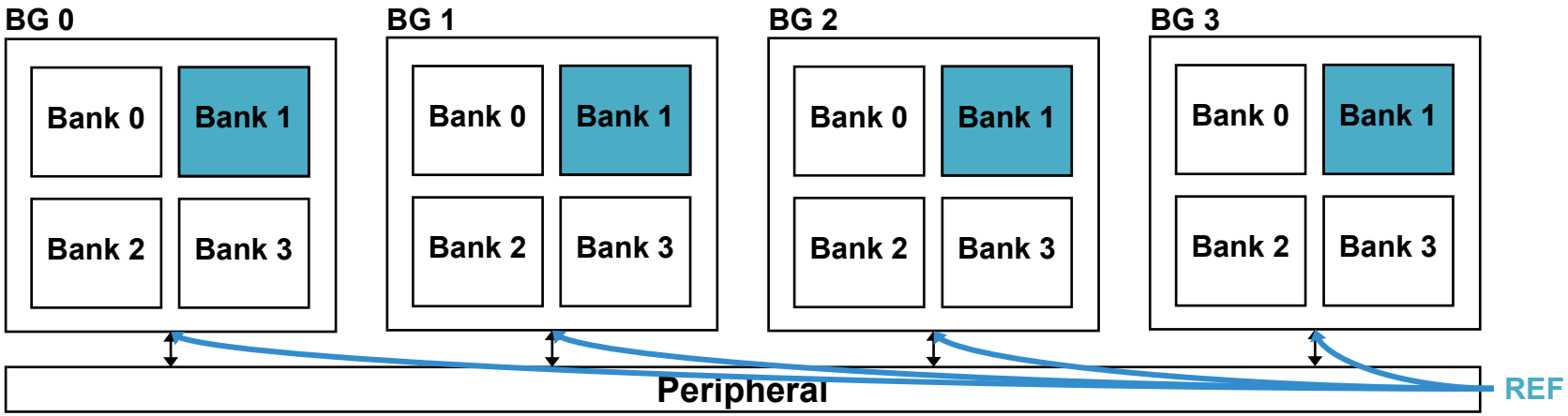
System overview

- **Hardware request processing path (including RMW)**
  - Hardware pipeline for processing requests
  - Major difference compared to NAND controller
- **DRAM subsystem**
  - Path for accessing AIT in DRAM requested by **AIT Manager** or **wear-leveling**
  - DRAM controller is simply instantiated using uCMD engine
- **RMW**
  - Read data as PCM access granularity (≥128 B) + partial update (64B)
  - Merge write data of same address / directly serve read request



More features in uCMD engine

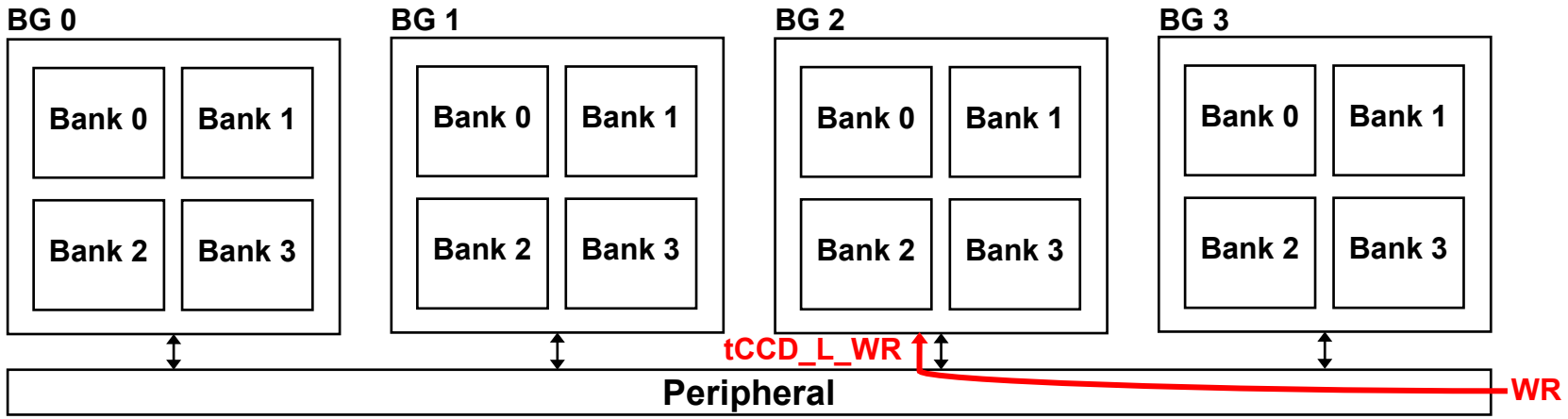
- DDR5 features are supported
  - **Same-bank refresh**: refresh banks with same index, across all bank groups



  - Diverged timing parameter of **tCCD_L_WR**
    - **Def.**: Turn-around latency of two successive write commands in same bankgroup
    - tCCD_L_WR: RMW latency → if total prefetch length ≠ on-die ECC codeword length
    - tCCD_L_WR2: just-write (JW) latency → if total prefetch length == on-die ECC codeword length
- Low-power features are supported
  - Self-refresh, fast- and slow-exit powerdown modes
  - Facilitate future low-power research in a DRAM system

More features in uCMD engine

- DDR5 features are supported
  - **Same-bank refresh**: refresh banks with same index, across all bank groups



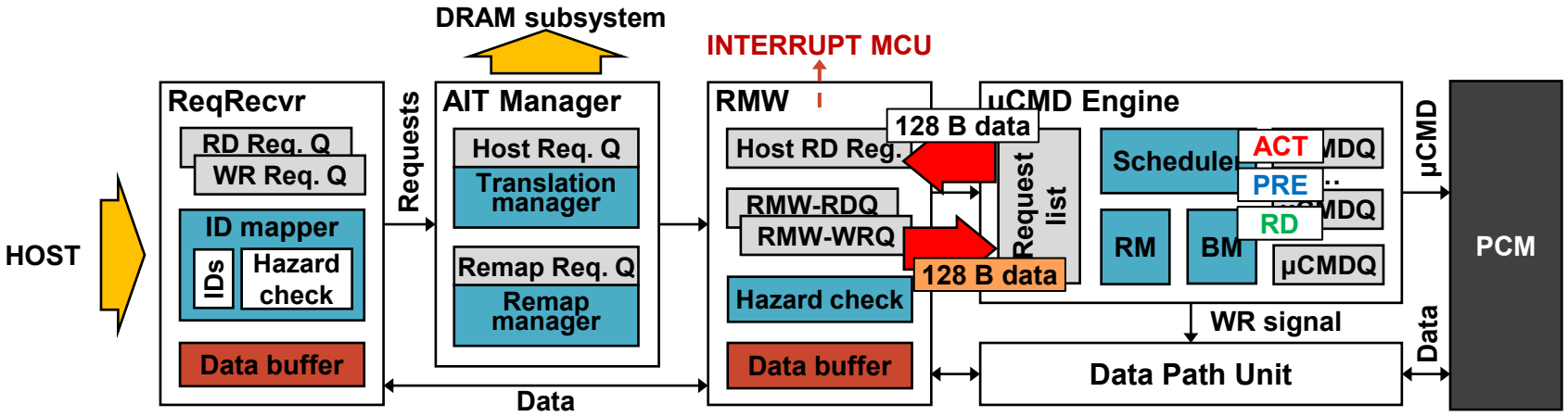
  - Diverged timing parameter of `tCCD_L_WR`
    - **Def.**: Turn-around latency of two successive write commands in same bankgroup
    - `tCCD_L_WR`: RMW latency → if total prefetch length ≠ on-die ECC codeword length
    - `tCCD_L_WR2`: just-write (JW) latency → if total prefetch length == on-die ECC codeword length
- Low-power features are supported
  - Self-refresh, fast- and slow-exit powerdown modes
  - Facilitate future low-power research in a DRAM system

Outline

- Background & Goal
- Motivation
- **Our simulator: PCMCsim**
  - System overview
  - **Execution flow**
  - Optimization: AIT Manager Prefetcher
- Evaluation
  - Accuracy validation
  - Takeaway messages
- Conclusion

Execution flow: hardware pipeline

- Request Receiver (ReqRecvr) stages host requests
  - Nearest interface for the host
  - Allocate request ID & Hazard handling (RAW, WAW, WAR)
- Two functionalities in AIT Manager
  - Function 1: address translation by accessing DRAM subsystem
  - Function 2: update AIT on DRAM subsystem & generate swap requests to PCM-side
- RMW
  - Reads data with PCM access granularity → partial update → write back
  - Interrupt MCU if write counts > THRESHOLD
- uCME engine ➔ decompose ***request*** into ***micro-commands***

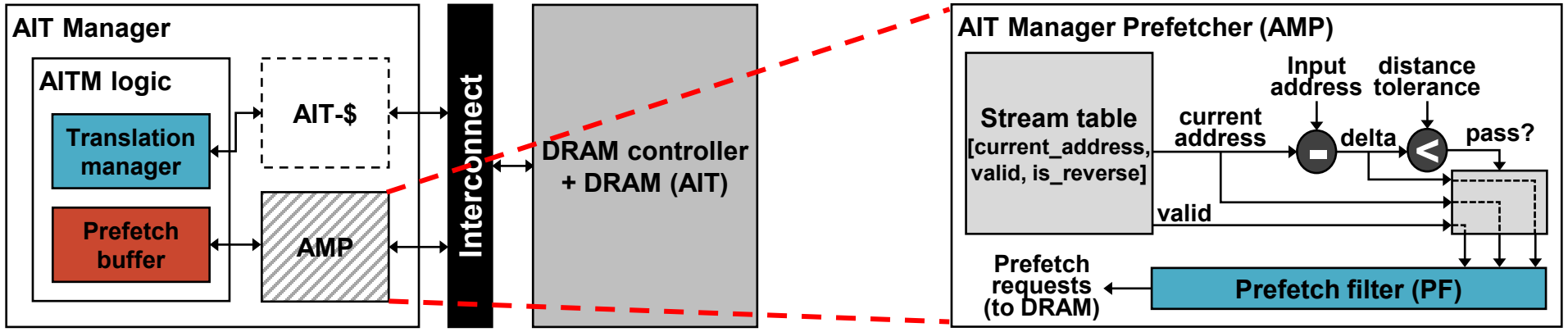


Outline

- Background & Goal
- Motivation
- **Our simulator: PCMCsim**
  - System overview
  - Execution flow
  - **Optimization: AIT Manager Prefetcher**
- Evaluation
  - Accuracy validation
  - Takeaway messages
- Conclusion

Optimization: AIT Manager Prefetcher

- Accessing AIT in DRAM incurs significant performance overhead
  - Contributes more than **20% of total latency** (shown later)
- AIT Manager Prefetcher ➔ reduce the access latency
  - Stream table → detects stream pattern
    - delta = input_address - current_address
    - |delta| ≤ distance tolerance → update current_address + pass previous entry value to PF
    - |delta| > distance tolerance → replace the LRU entry with input_address
  - Prefetch filter → prevents generating same prefetch requests
    - valid==0: part of new stream pattern → prefetch new stream pattern
    - valid==1: prefetched previous "stream" → prefetch next stream pattern

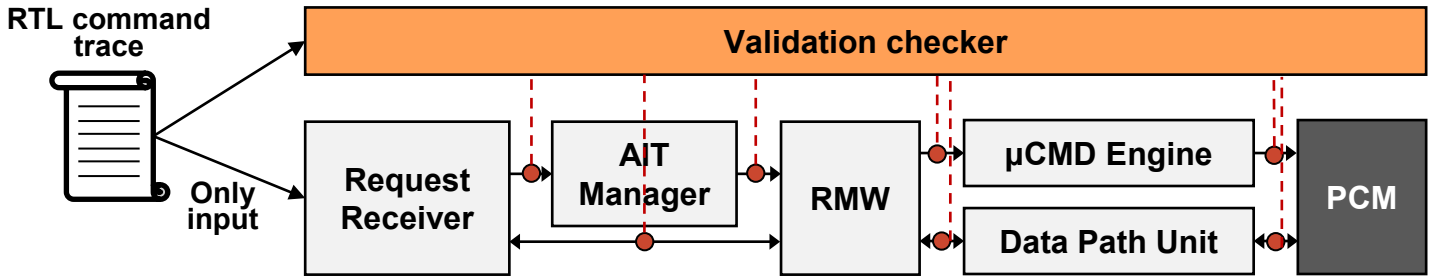


Outline

- Background & Goal
- Motivation
- Our simulator: PCMCsim
  - System overview
  - Execution flow
  - Optimization: AIT Manager Prefetcher
- **Evaluation**
  - **Accuracy validation**
  - Takeaway messages
- Conclusion

Accuracy validation

- Validated cycle accuracy and functional accuracy
- Inject hardware RTL command trace to PCMCsim and validation checker
  - Info: > 2K commands with read/write ratio of 1.25 and various access patterns
  - Functional accuracy → probed commands between two modules
  - Cycle accuracy → compared simulated cycles and timestamps in RTL trace



- **Functional error: 3.16%; cycle error: 13.6%**

Outline

- Background & Goal
- Motivation
- Our simulator: PCMCsim
  - System overview
  - Execution flow
  - Optimization: AIT Manager Prefetcher
- **Evaluation**
  - Accuracy validation
  - **Takeaway messages**
- Conclusion

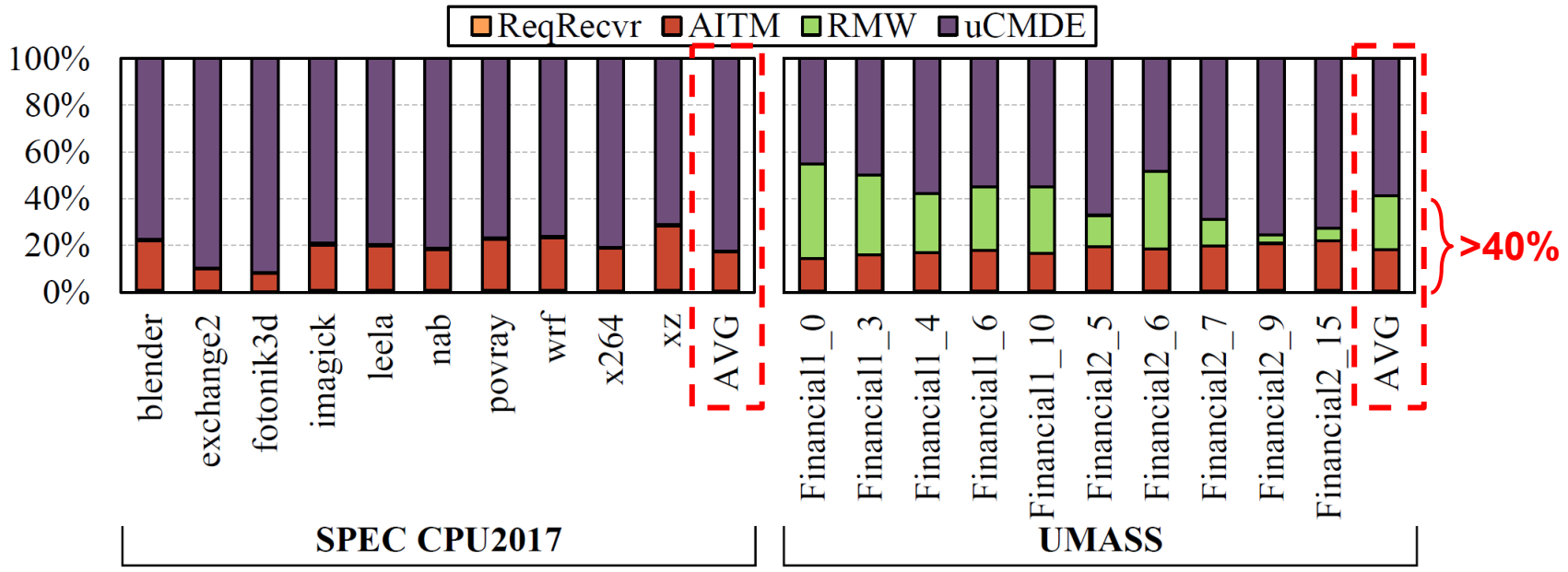
Methodology

- Simulator
  - gem5 (for memory trace extraction), PCMCsim (as a main memory)
- Workloads
  - SPEC CPU 2017 memory trace (extracted from gem5)
  - Umass OLTP Application I/O (UMASS) traces
- System parameters
  - gem5: Intel i7-7700
  - Baseline in PCMCsim (see below)

ReqRecvr		single queue, 128 request IDs available
AITM	**logic**	2-entry command queue
	DRAM-AIT	DDR4-2666, 4 GB, ×16 device, 64 B line 1 rank, 2 bank groups, 4 banks/bank group, tRAS=35 ns, tRCD=14.25 ns, tWR=15 ns, tCL=14.25 ns, tRP=14.25 ns,
RMW		2-entry data buffer for read-modify-write
uCMDE (PCM)		DDR4-2666, 128 GB, ×16, 128 B line 1 rank, 2 bank groups, 4 banks/bank group, tRAS=100 ns, tRCD=100 ns, tWR=150 ns, tCL=14.25 ns, tRP=14.25 ns

Observation 1: latency breakdown

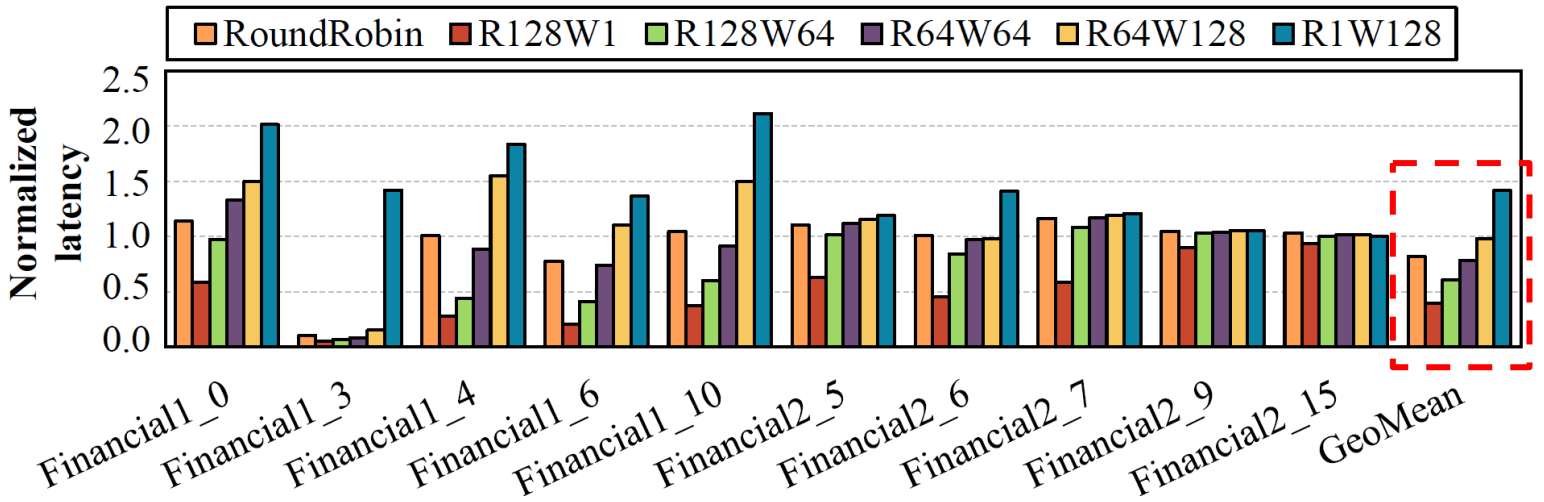
- Accessing AIT contributes 20% of total latency for both workloads
- SPEC CPU 2017 traces
  - **No write requests** → negligible RMW overhead
- UMASS
  - Many write requests → RMW contributes 25% of total latency



Message 1: architectural bottlenecks in a modern PCM controller newly come from AIT and RMW

Observation 2: Request Receiver

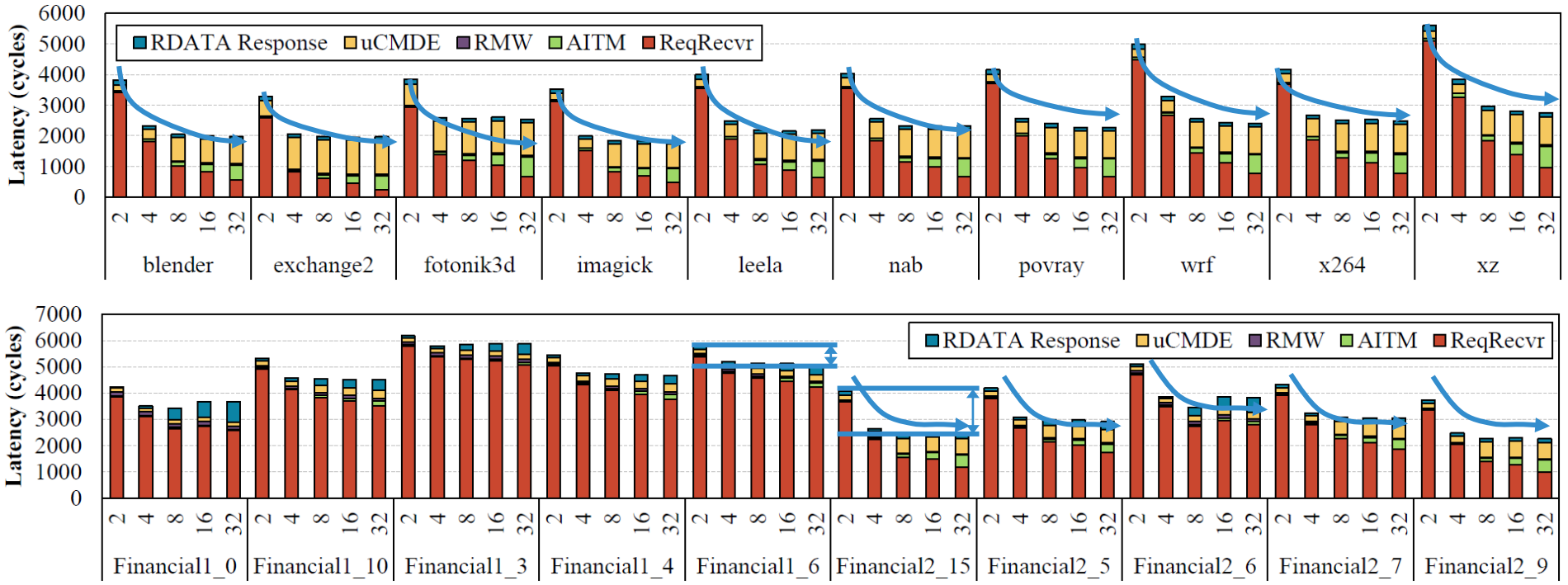
- Sweeping watermarks of read & write queues in Request Receiver
  - R**x**W**y**: issue WR after **x** RDs; issue RD after **y** WRs
  - Round Robin (R1W1): alternately issue read request and write request
- No latency variation for SPEC CPU 2017 → **No write requests**
- Latency increases (i.e., lower QoS) as the priority of read request lowers
- **No effects on throughput** → determined by WAWs in RMW than ReqRecvr



Message 2: watermark configuration in Request Receiver determines QoS rather than throughput for write-intensive workloads

Observation 3: AIT Manager

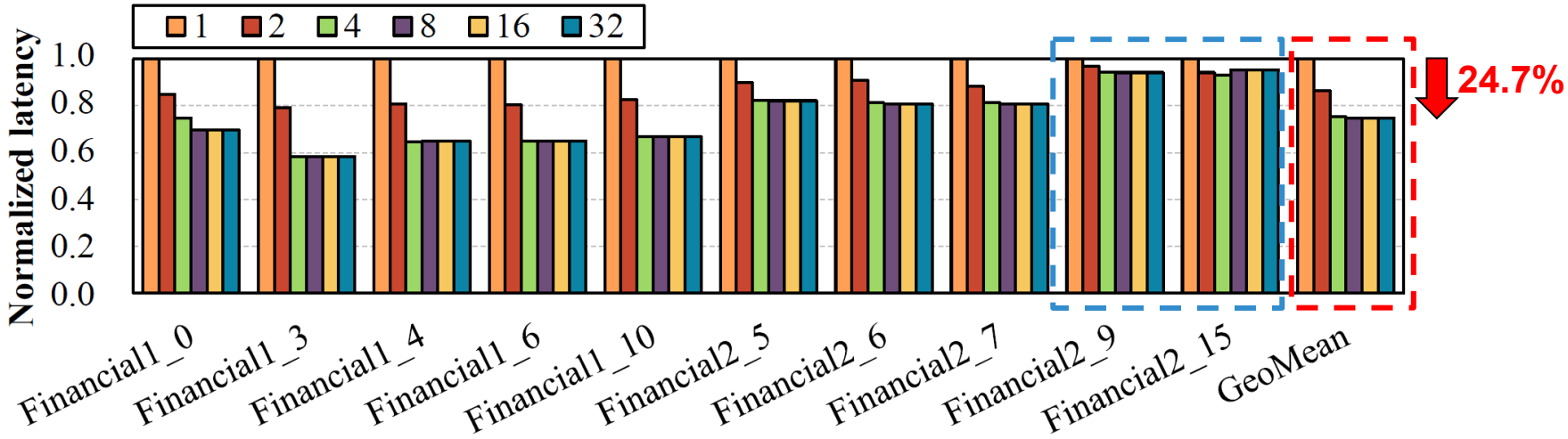
- Sweep host request queue size in AIT manager
- Larger queue → lower latency due to **reduced backpressure** on the pipeline
- More improvement variances for UMASS due to R/W ratio → more details in paper
- Increasing queue size amplifies the latency of modules



Message 3: host request queue size in AIT manager, which is cost-effective at 8, is a key parameter to reduce pipeline backpressure on Request Receiver

Observation 4: RMW

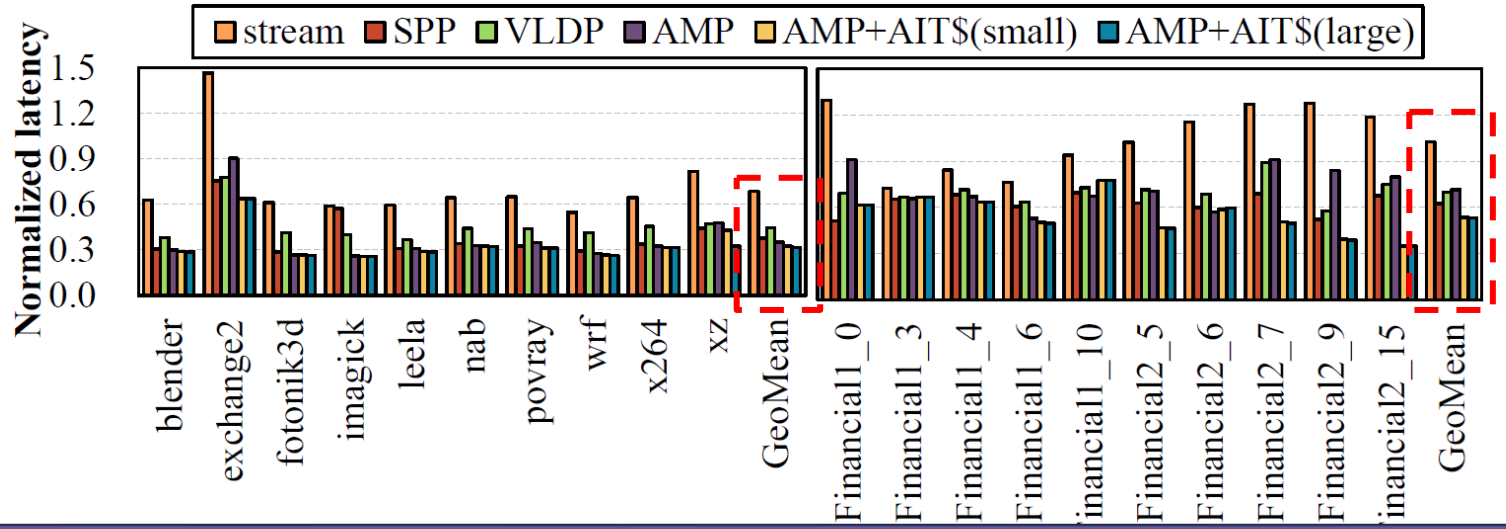
- Sweep data buffer size in RMW
- No latency variation for SPEC CPU 2017 → **No write requests**
- Larger data buffer → **lower latency**
- Some workloads → **small QoS improvement** due to relatively low merge ratio



Message 4: for write-intensive workloads (e.g., UMASS), increasing the number of data buffer entries in RMW significantly improves the performance because of merge operation in RMW

Observation 5: prefetcher performance

- Performance of different prefetcher schemes
  - Previous studies: stream (i.e., next-line), SPP, VLDP
  - AMP+AIT$(small): AMP and 1 KB of AIT cache
  - AMP+AIT$(large): AMP and 64 KB of AIT cache
- Different prefetchers are suitable for different workloads
  - AMP is better for SPEC CPU 2017 → highest prefetch accuracy
  - AMP is worse for UMASS → lower prefetch accuracy for detecting stream patterns



Message 5: applying prefetcher schemes in AIT manager significantly improves QoS (up to 68.5%), but it is recommended to combine different prefetch schemes for general applications (i.e., no silver bullet)

Outline

- Background & Goal
- Motivation
- Our simulator: PCMCsim
  - System overview
  - Execution flow
  - Optimization: AIT Manager Prefetcher
- Evaluation
  - Accuracy validation
  - Takeaway messages
- **Conclusion**

Conclusion

- PCMCsim is a new simulator for modern PCM controller
  - AIT (including DRAM subsystem), RMW, hardware processing path are included
  - High simulation accuracy
  - Supports newest standard (DDR5) and low-power features for general-use

- Characterize the performance of modern PCM controllers
  - Provide insights for modern PCM controllers
  - AIT and RMW contribute majority of the latency

Thank you!

- For information…
  - Q&A / email: hklee@capp.snu.ac.kr
  - More details in paper
  - Code is available at: https://github.com/harrylee365/pcmcsim_public