# Accelerating Big Data Applications: Next Generation Memory/Storage Systems

**Chun-Feng Wu (吳俊峯)**

**Assistant Professor, National Yang Ming Chiao Tung University**

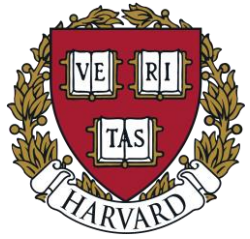
cfwu417@cs.nycu.edu.tw

https://cfwu417.github.io

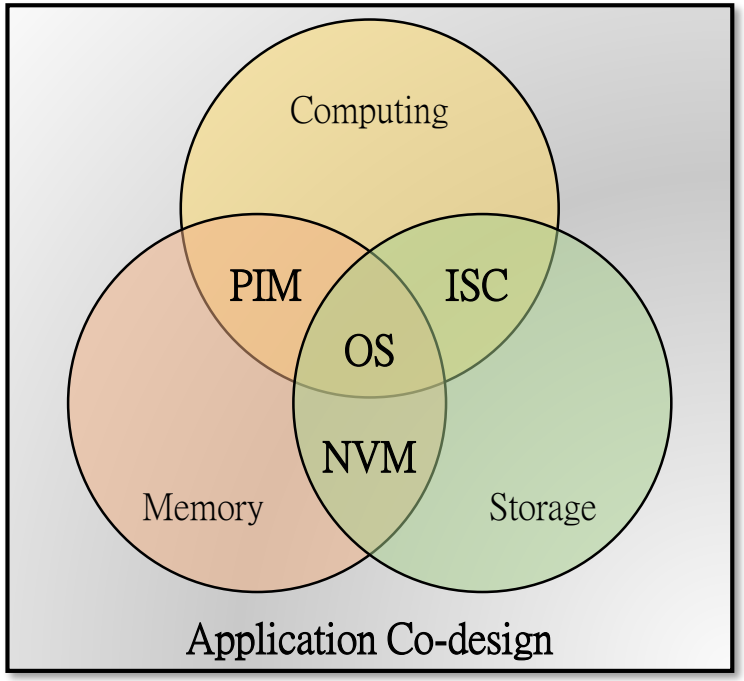
# Personal Information

- **Chun-Feng Wu (吳俊峯)**
  Assistant Professor @ Department of Computer Science,
  National Yang Ming Chiao Tung University
  Prior: Postdoc @ Harvard University
  - Website: https://cfwu417.github.io
  - **O**perating **S**ystem and **C**omputer **AR**chitecture (OSCAR) Lab (EC119)

Cooperation


- **Research Interests**
  - Operating System (OS)
  - Processing-in-Memory (PIM) (AI Accelerator)
  - In-Storage Computing (ISC)
  - Co-design Applications with Memory/Storage Devices
  - Emerging non-volatile memory (NVM)
    - STT-RAM, ReRAM, PCM, Flash Memory (SSD), SMR (HDD)



# Education and Experience

**09/2014 - 06/2016**

- **Master** at Department of Computer Science, National Tsing Hua University
  - Advisor: Prof. Yeh-Chin Chung (鍾葉青教授)
  - Master Thesis: *Hybrid mechanisms to improve write scenarios for cloud storage services*

**09/2016 - 06/2021**

- **PhD** at Department of Computer Science and Information Engineering, National Taiwan University
  - Advisor: Prof. Tei-Wei Kuo (郭大維教授)
  - Ranking in class: Rank #2, GPA: 4.17
  - Dissertation: *Support to Huge Main-Memory Extension with Non-Volatile Memory*

**08/2017 - 08/2021**

- **Research Assistant** at Institute of Information Science, Academia Sinica
  - PI: Prof. Yuan-Hao Chang (張原豪教授)

**08/2021 – 07/2022**

- **Post-Doctoral Scholar** at Department of Computer Science, Harvard University, Cambridge, USA
  - Lab PI: Prof. David Brooks & Prof. Gu-Yeon Wei
  - Website: https://vlsiarch.eecs.harvard.edu/people

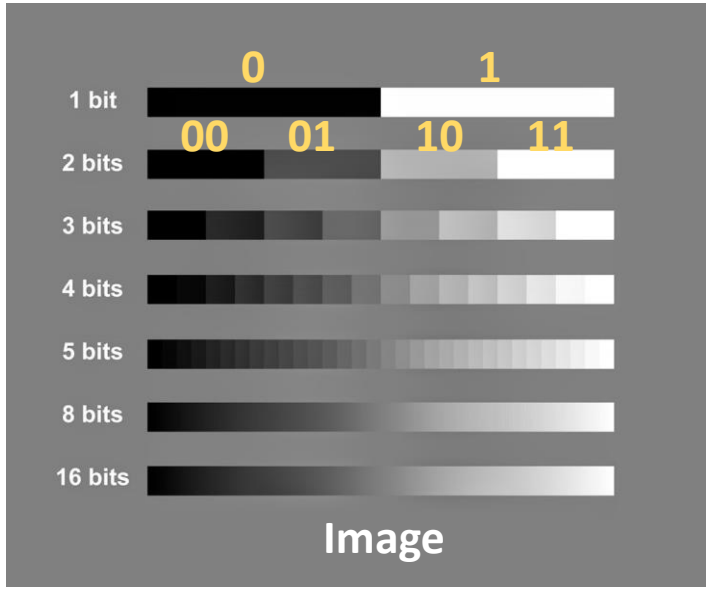
Cooperate with SAMSUNG and Meta

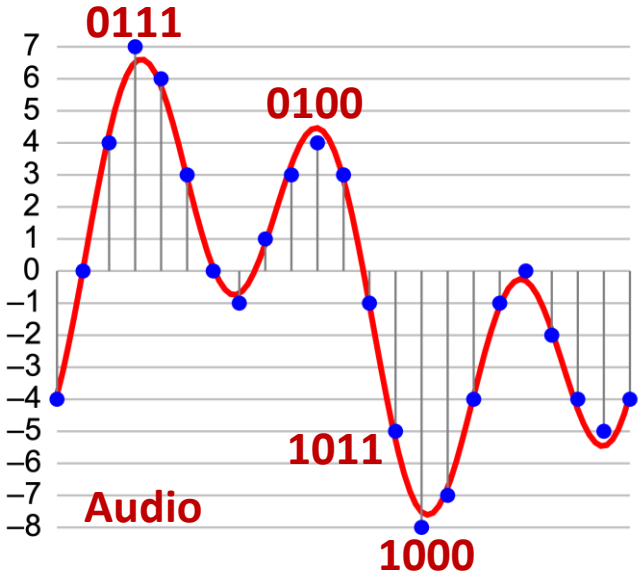
# Agenda

- Background: Data & Storage/Memory
- Challenge of Big Data Era
- Next-Generation Memory/Storage System
  - Processing-in-Memory
  - Hybrid Memory
  - Co-designing Application and Devices
- Case Study: Co-designing Recommendation Systems with SSDs
- Future Plan
- Q&A

# Data & Memory/Storage Devices

- What is data: A sequence of 0 and 1.
- What are storage/memory devices: Devices consist of material which can switch between 2 states (i.e., 0 and 1) stably.
  - Magnetic Field (Parallel vs Anti-Parallel): HDD
  - Voltage (High vs Low): SSD
  - Capacitor (Charged or not): DRAM
  - Resistance (High vs Low): ReRAM, PCM, STT-MRAM



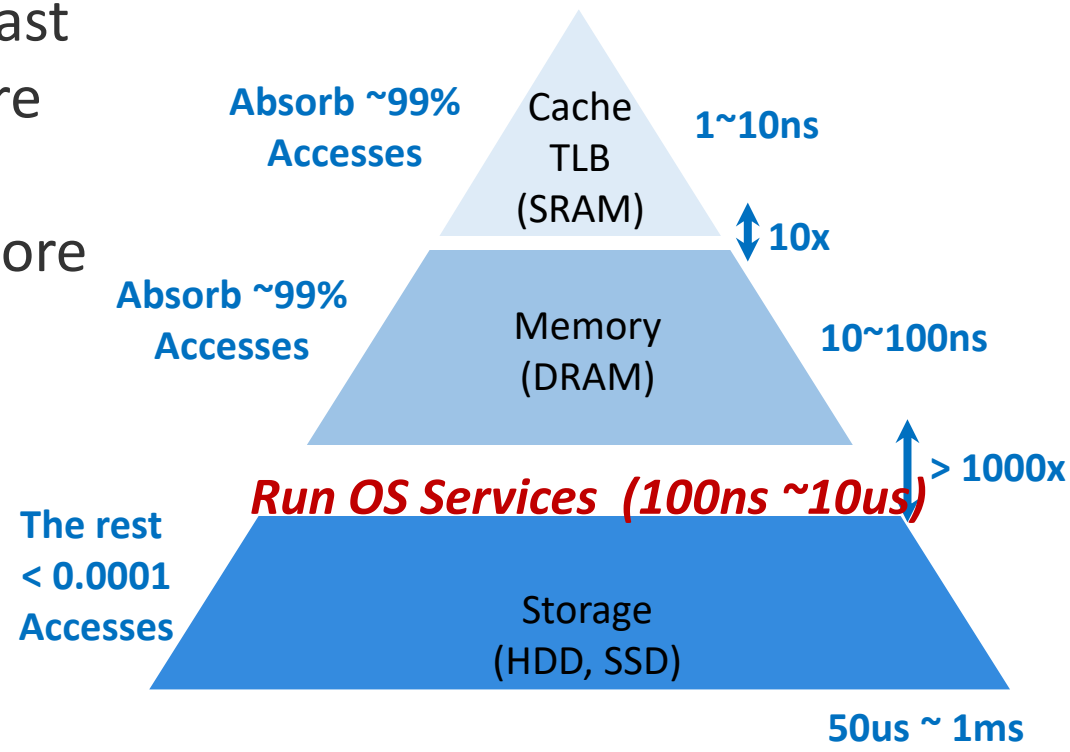

[1] https://www.lumenera.com/blog/bit-depth
[2] https://en.wikipedia.org/wiki/Audio_bit_depth

# What is Storage & Data

- What is data: A sequence of 0 and 1.
- What are storage/memory devices: Devices consist of material which can switch between 2 states (i.e., 0 and 1) easily.
  - Magnetic Field (Parallel vs Anti-Parallel): HDD
  - Voltage (High vs Low): SSD
  - Capacitor (Charged or not): DRAM
  - Resistance (High vs Low): ReRAM, PCM, STT-MRAM
- Volatile vs Non-Volatile : Retain the data, even when the power is off?
  - SSD is non-volatile, DRAM is volatile
- Access Granularity: Minimum read/write size for each device.
  - Byte-addressable (Cacheline 64B): DRAM, ReRAM, PCM, STT-MRAM
  - Flash Page (4KB ~ 16KB): SSD
  - Sector (512B ~ 4KB): HDD

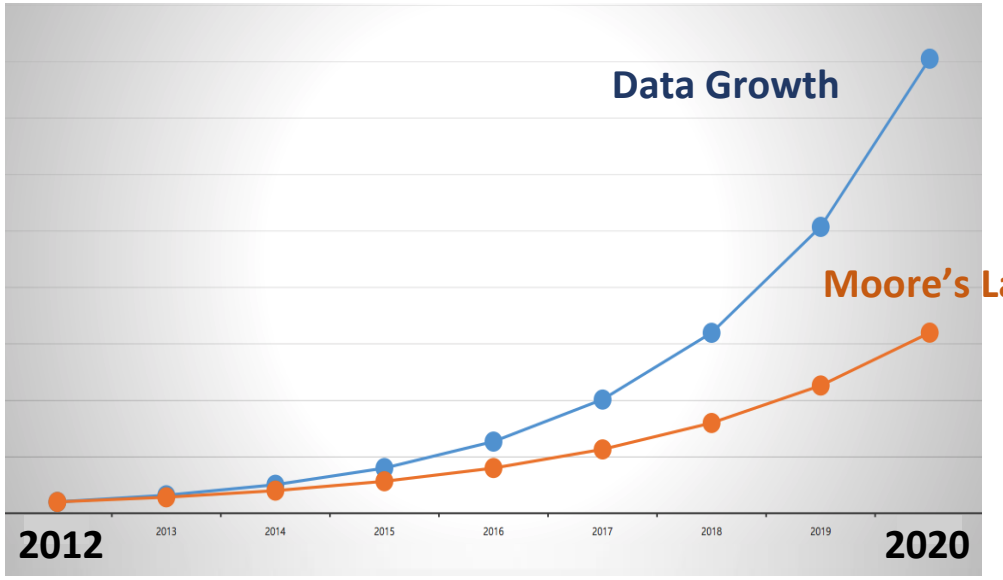
# Hierarchy: Cache vs Memory vs Storage

- Memory Hierarchy: Balance access time and cost.
  - CPU cache is super fast. Memory devices are at least 10x slower than CPU cache. But storage devices are even 1000x slower.
  - Locality: Cache can absorb most accesses but is more expensive.
- OS execution time is slower than memory accesses but faster than storage accesses.
  - Get data from memory: CPU loads/stores.
  - Get data from storage: Shall execute OS.
    (Ex: page fault handler or R/W system calls)



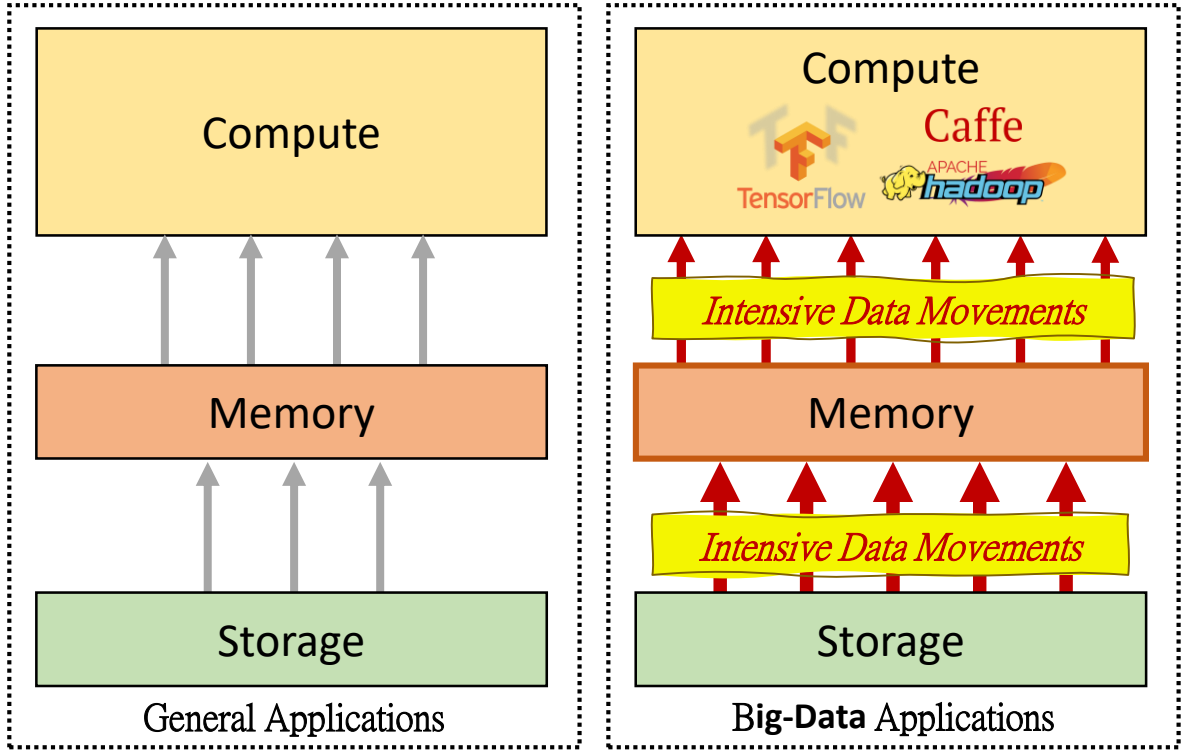
# Challenges of von-Neumann Machines

- The **data growth** surpasses the **memory capacity growth**.
- **Intensive data movements** among computing, memory, and storage units may degrade system performance.




CIO Summit, 2017

**Data Growth vs. Moore's Law**




**Challenges of von-Neumann Machines**

# Emerging Memory and Storage Devices

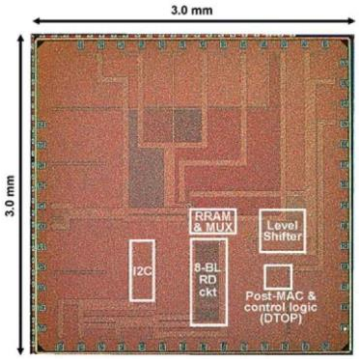

NVMe M.2 SSD


NVMe U.2 SSD


NVMe PCIe SSD

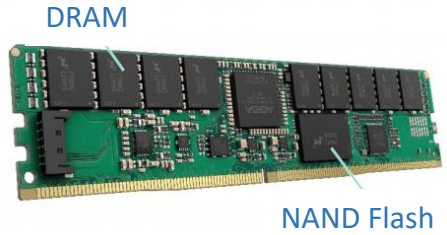

Intel Optane SSD

Intel Optane DC
Persistent Memory

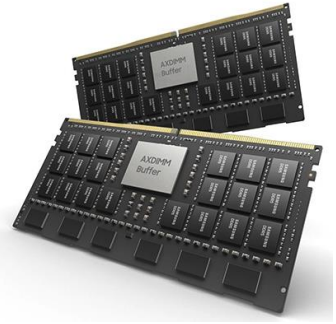



TSMC PIM Chips


Samsung Z-NAND SSD




HPE NVDIMM




Samsung AXDIMM

**High Throughput**
*3.98 GB/s*

**Ultra-Low Latency**
*~ 5 us*

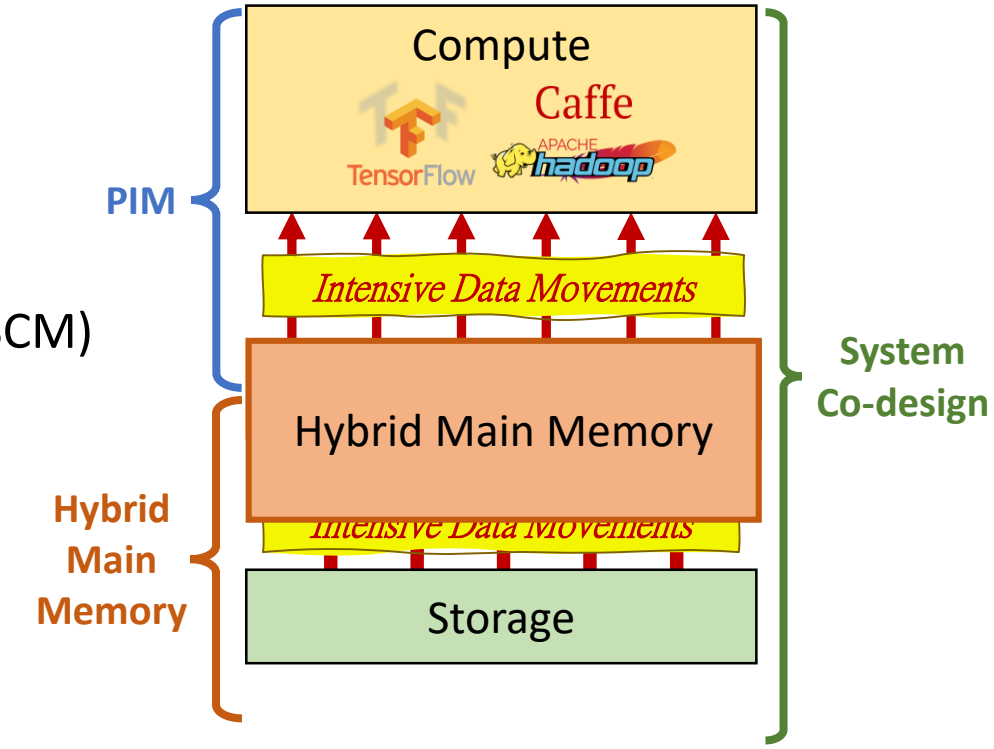
**Storage-Class Memory**

**Near-Data Computation**

**2011** — **2017** — **Now**

# Next-Generation Computing Systems

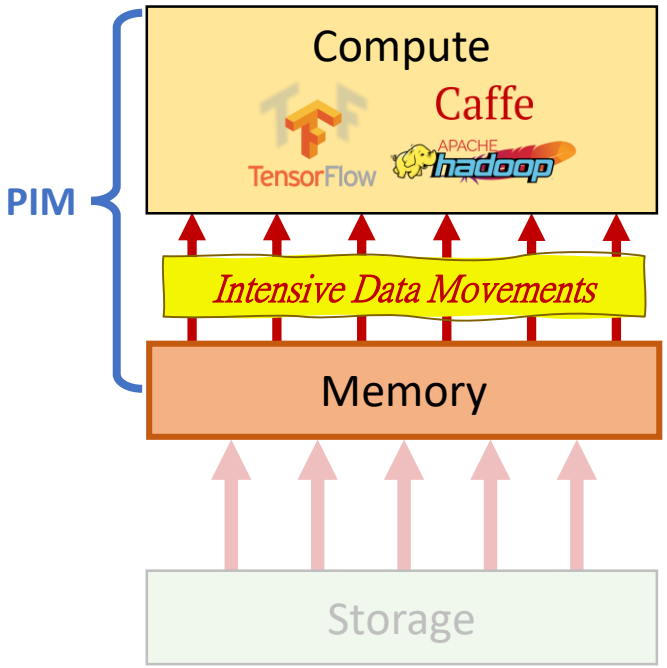
- To deal with performance bottleneck caused by intensive data movements, we present a next-generation computing system based on emerging devices.
- **Technique highlights and challenges**
  - ✓ Processing-in-Memory (PIM)
    - Adaptive data allocation to deal with scalability issue
    - Re-arrange graph indexing to deal with utilization issue
  - ✓ Hybrid Main Memory
    - Joint management of CPU and Storage-Class Memory (SCM) devices to break down the great memory wall
    - Re-design OS management to enjoy the benefit of ultra-low-latency and high bus throughput
  - ✓ System Co-design
    - Device-friendly application design to enhance device lifetime and improve quality-of-service
    - Today, we will focus on this part at the last 30 minutes

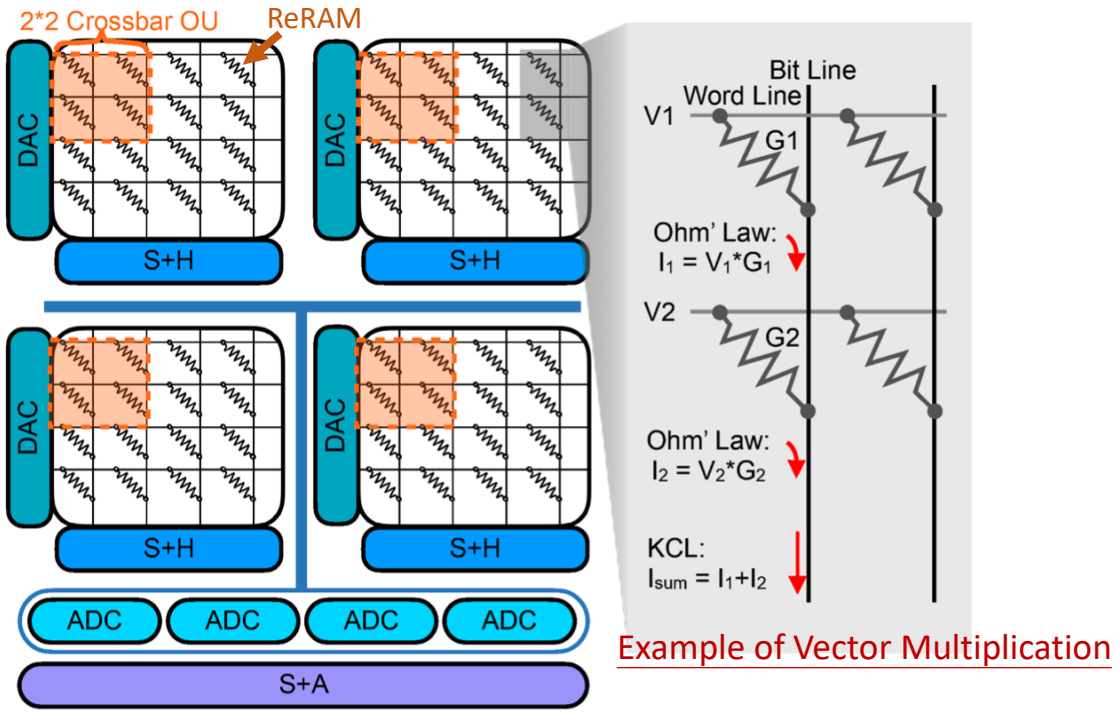


MXIC
Macronix International Co., Ltd.

# Processing-in-Memory (PIM) (1/3)

- Before being processed, data movements are unavoidable between memory and compute units.
- Leverage **Processing-in-Memory (PIM)** technology to eliminate data movements.




Example of Vector Multiplication

**ReRAM-based Crossbar Accelerator**

MXIC
Macronix International Co., Ltd.

# Processing-in-Memory (PIM) (2/3)

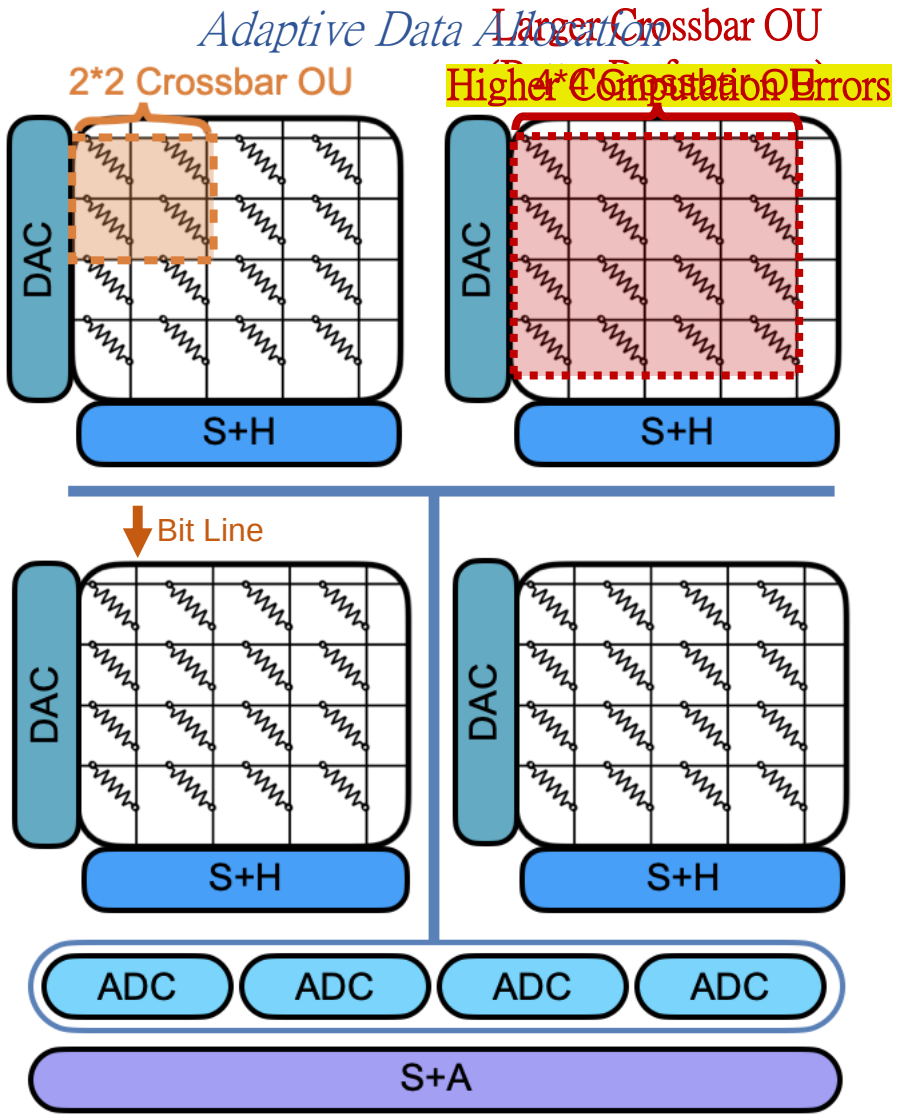
- Larger OU, Better Performance

  The performance of crossbar accelerator is proportional to the number of NVM cells in each operation unit (OU).

  However, larger OU also causes higher computation errors.

- **Adaptive Data Allocation**

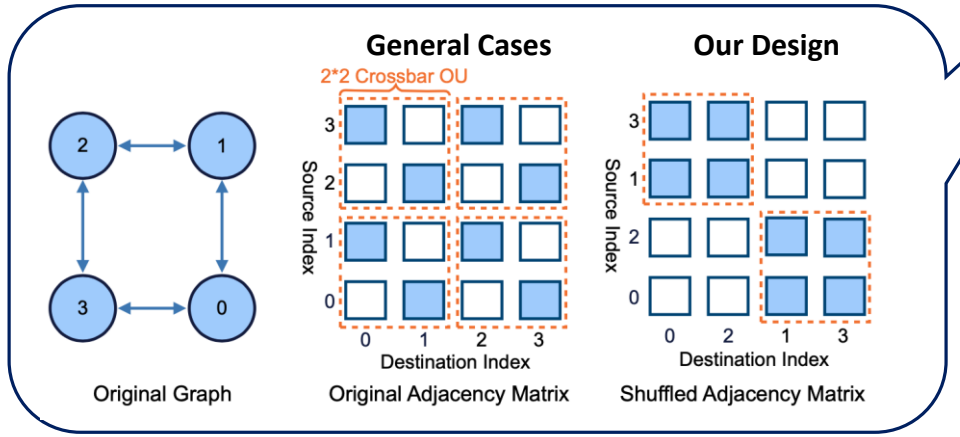
  To Deal with scalability issue, we allocate *small OUs for Most-Significant-Bit (MSB)* and *large OUs for Least-Significant-Bit (LSB)*

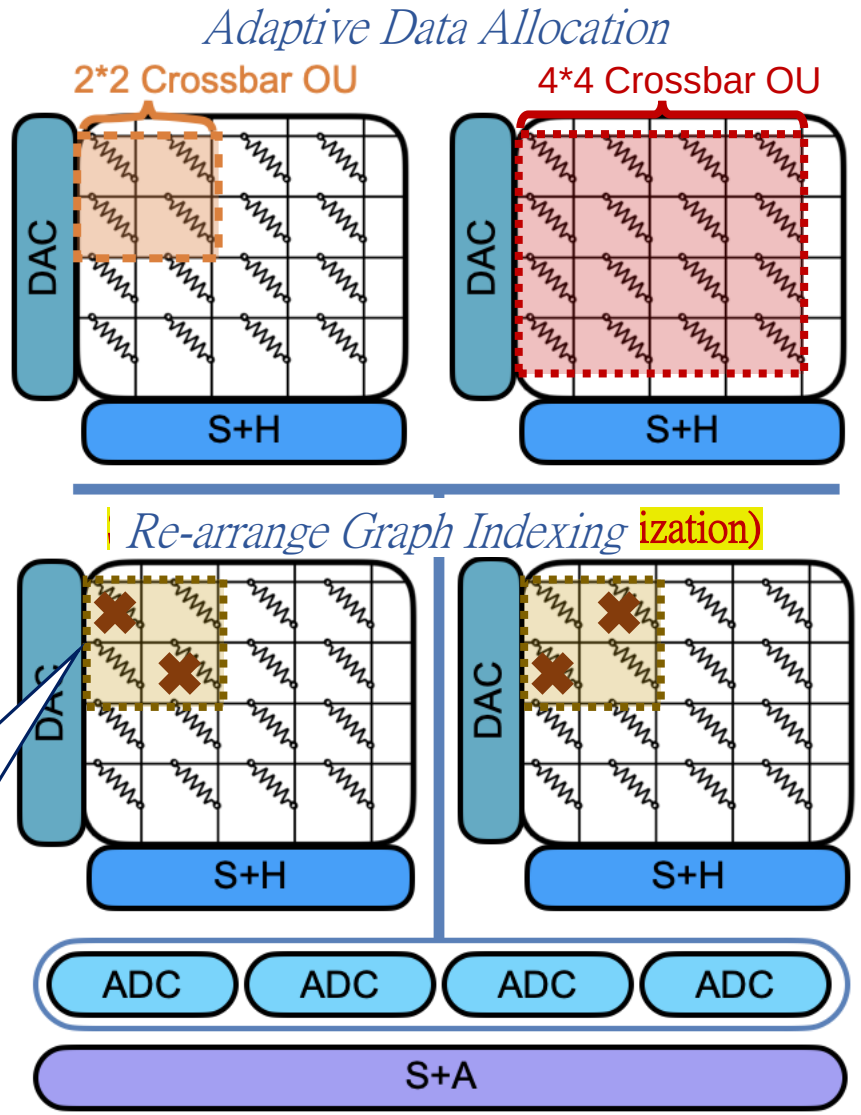


MACRONIX INTERNATIONAL CO., LTD.

# Processing-in-Memory (PIM) (3/3)

- Larger OU, Better Performance
  - The performance of crossbar accelerator is proportional to the number of NVM cells in each operation unit (OU).
  - However, larger OU also causes higher computation errors.
- **Re-arrange Graph Layout**
  - Graphs are usually stored as sparse matrix, which decreases the utilization of OUs.
  - To Deal with utilization issue, we *re-arrange graph indexing* to fit crossbar architecture.



MXIC
Macronix International Co., Ltd.

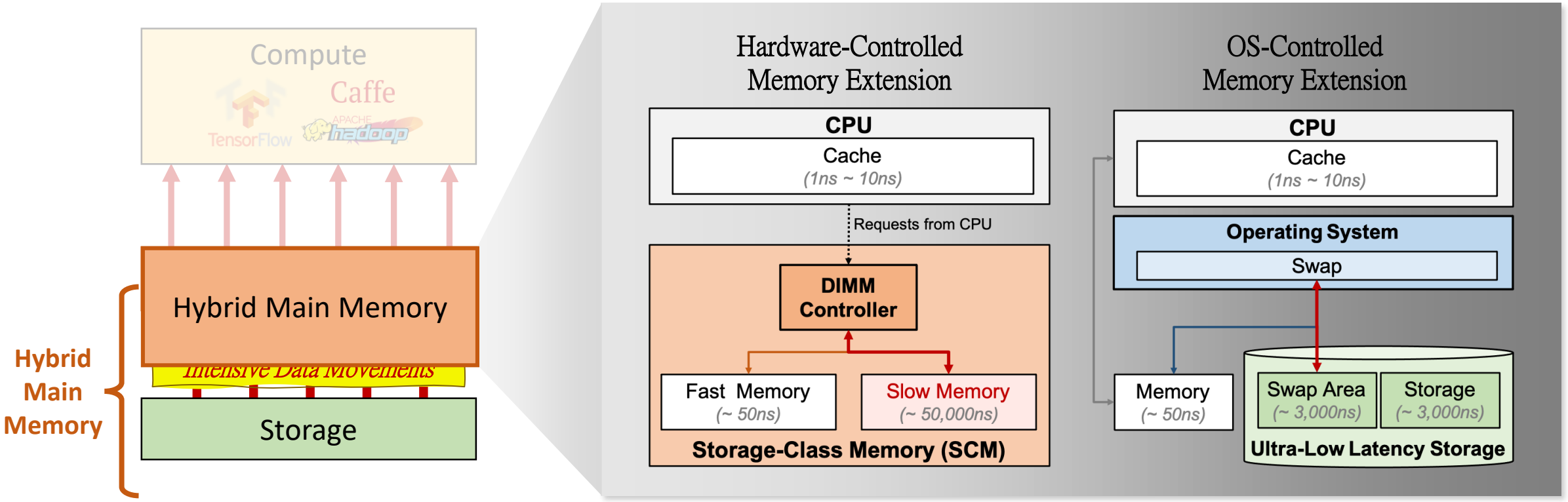
# Achievements of PIM Designs

- **Publications**
  - IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **(TCAD, 2020)** (Integrated with ACM/IEEE EMSOFT, Top Conference) (Work with Macronix)
  - ACM/IEEE International Symposium on Low Power Electronics and Design **(ISLPED, 2021)** (Top Conference) (Work with Macronix)
  - ACM/IEEE Design Automation and Test in Europe **(DATE, 2021)** (Work with TU Dortmund, Germany)
- **US Patent**
  - Neural Network Computation Method And Apparatus Using Adaptive Data Representation (Work with Macronix)
    - Patent Number: US 16,798,166
    - Publication Date: Oct. 1st, 2020

[1] https://www.samsung.com/semiconductor/newsroom/events/open-innovation-contest/#finalists

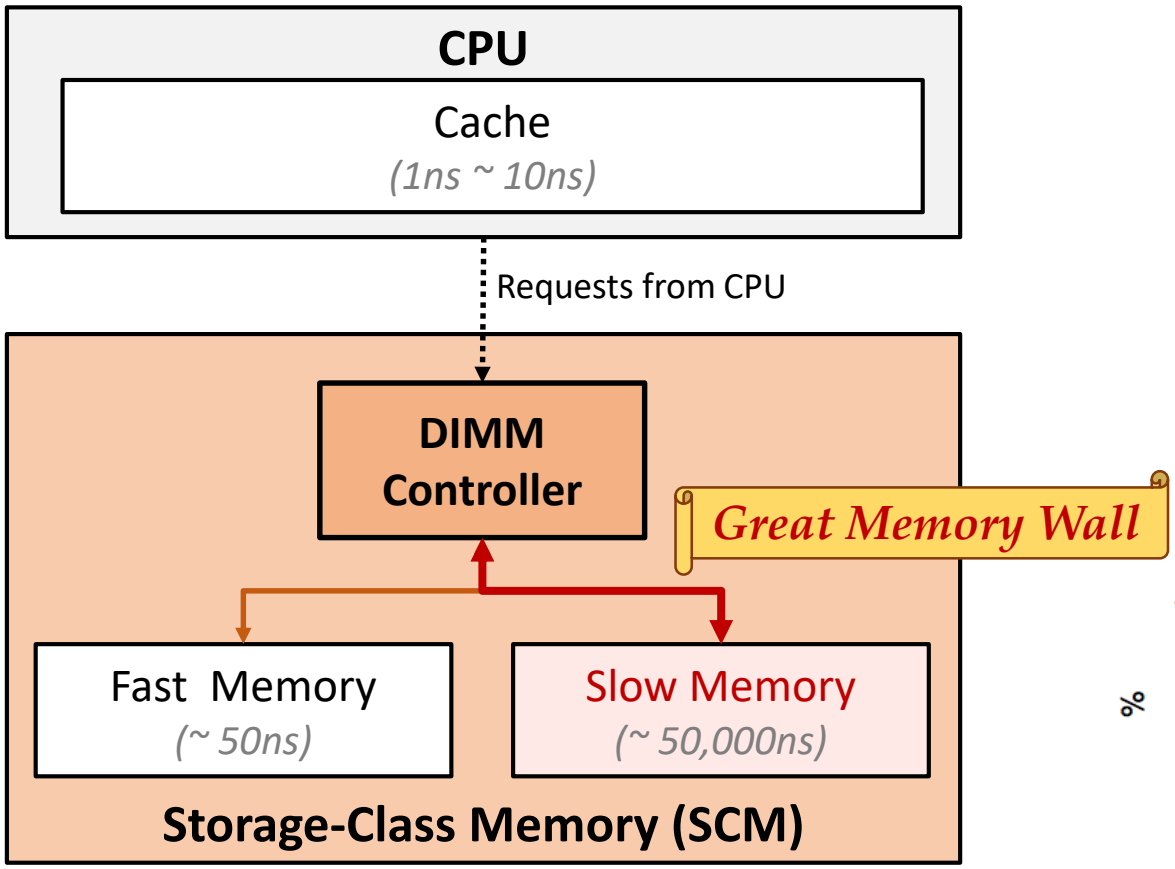
# Hybrid Main Memory (1/4)

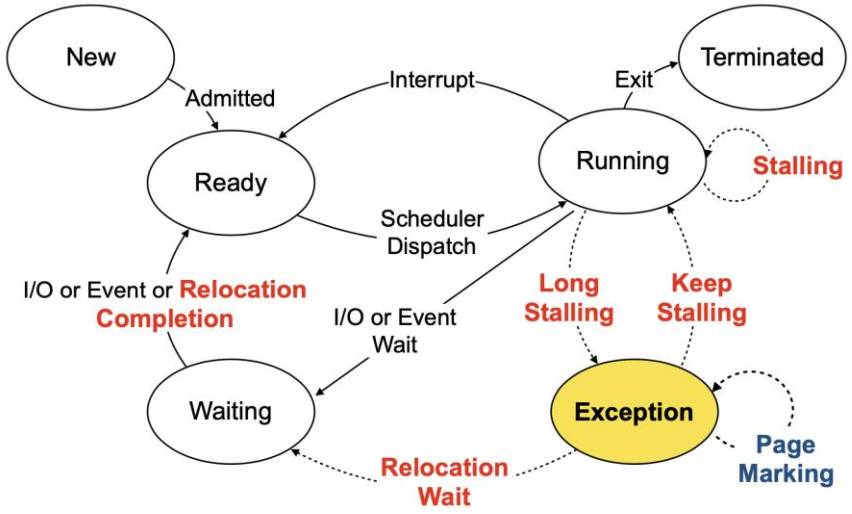
- Before being processed, data movements are unavoidable between storage and memory units.
- Realize low-cost high-performance main memory extension with **Hybrid Memory Devices**.



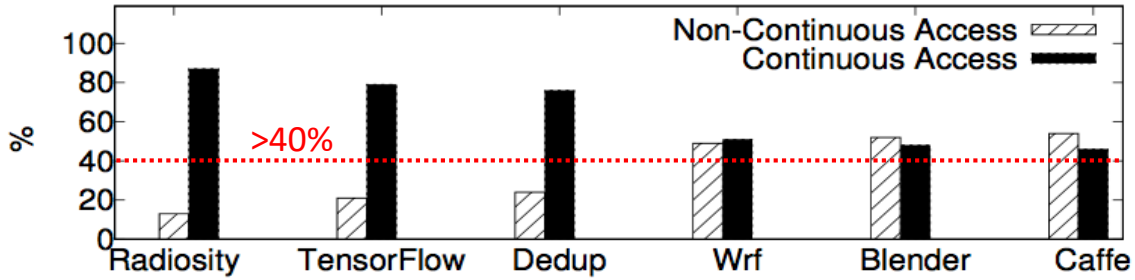
# Hybrid Main Memory (2/4)

- **Hardware-Controlled Memory Extension**
  - Joint Management of CPU and Storage-Class Memory (SCM) to break down the "great memory wall".



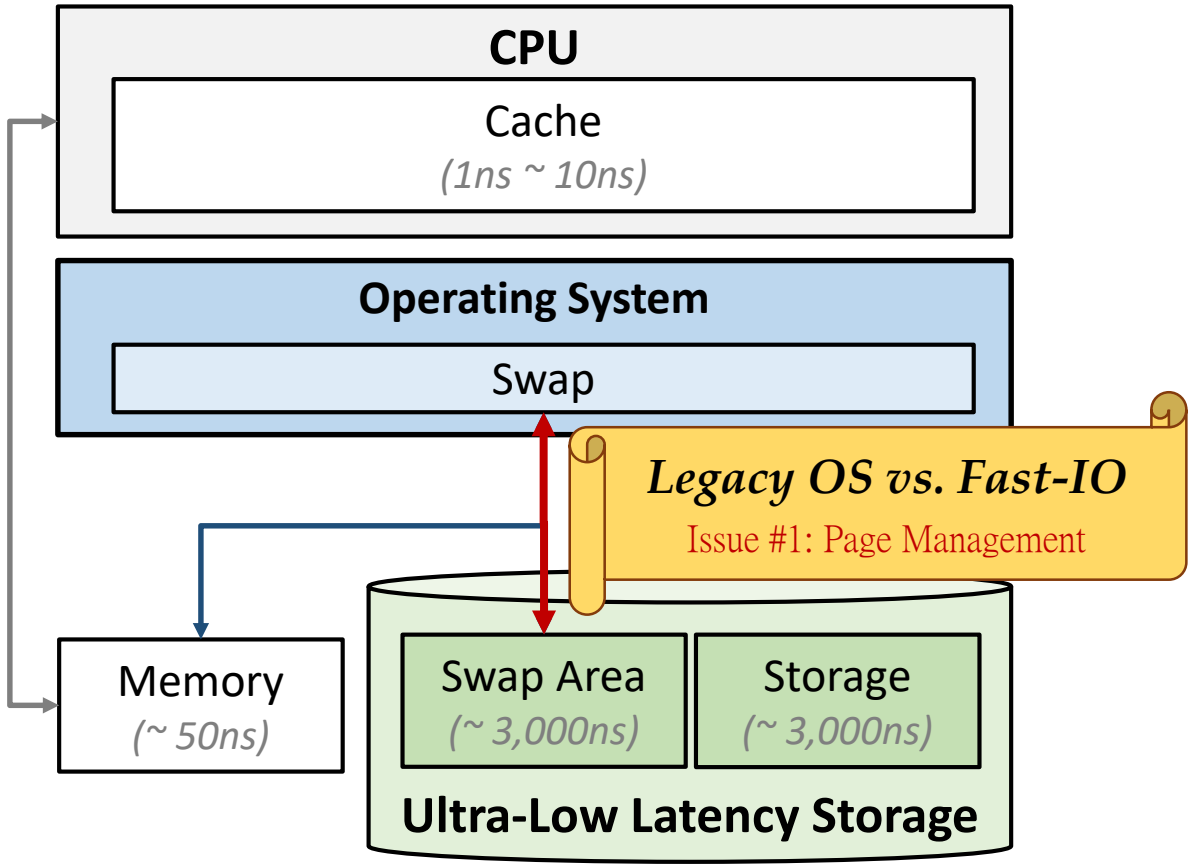

**Stall-aware Process State Diagram**

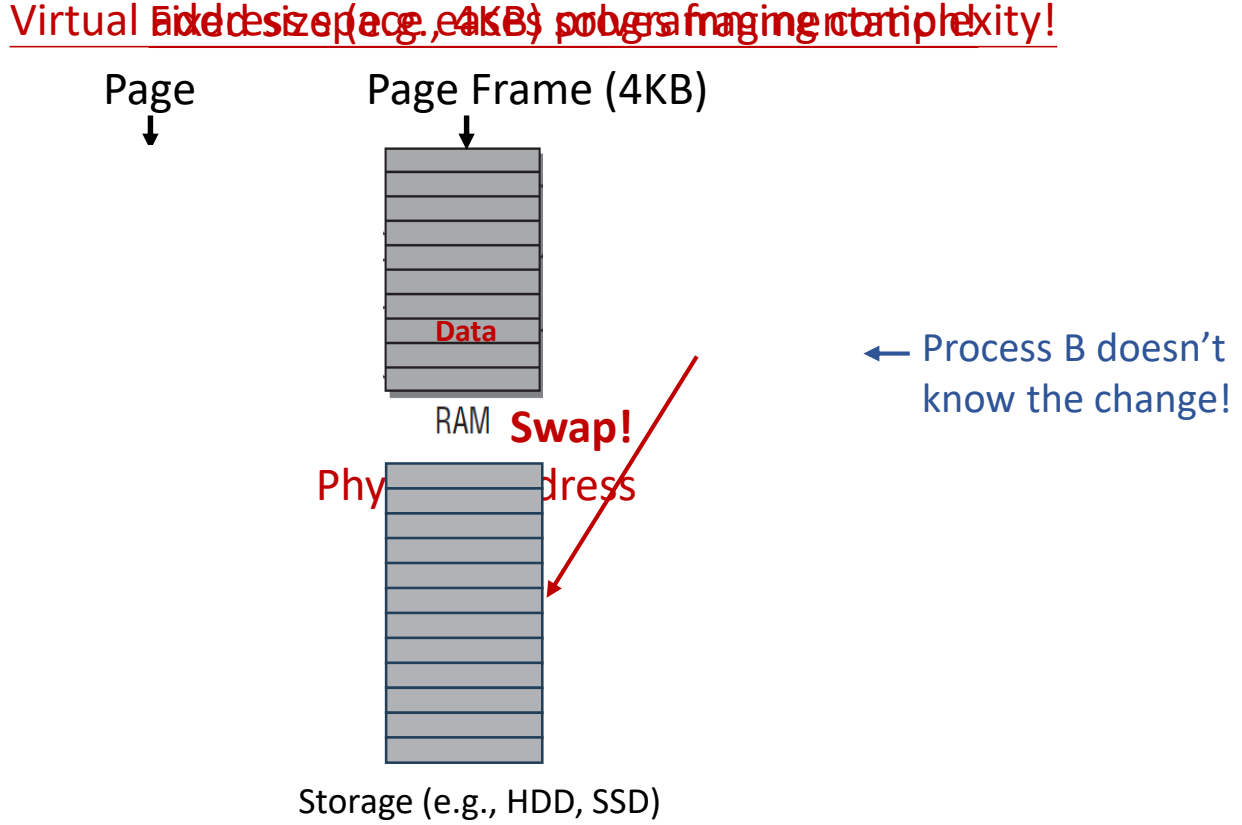



**40% of Memory Accesses Show High Spatial Locality**

# Hybrid Main Memory (3/4)

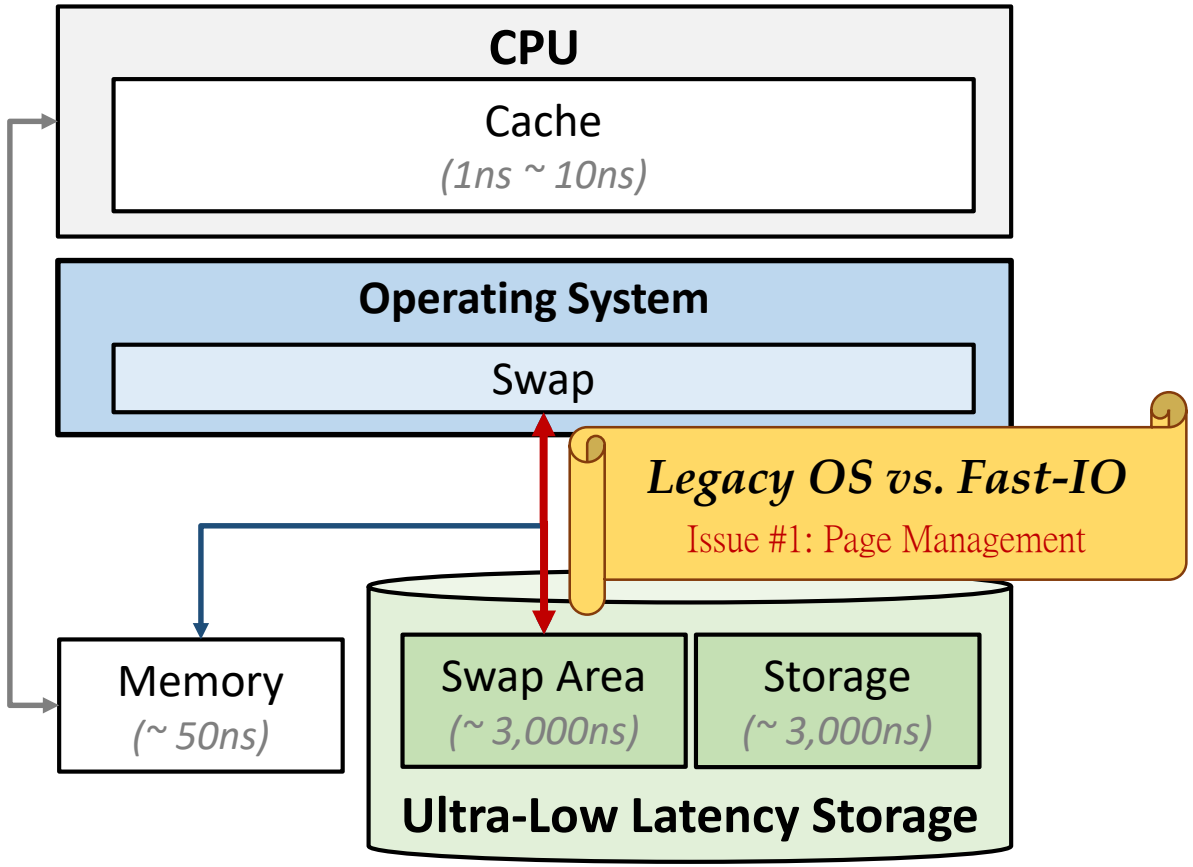
- **OS-Controlled Memory Extension**
  - Dynamic OS page management to enjoy the benefit of Ultra-Low-Latency (ULL) storage devices and high bus throughput. *(A rethinking of OS Paging)*

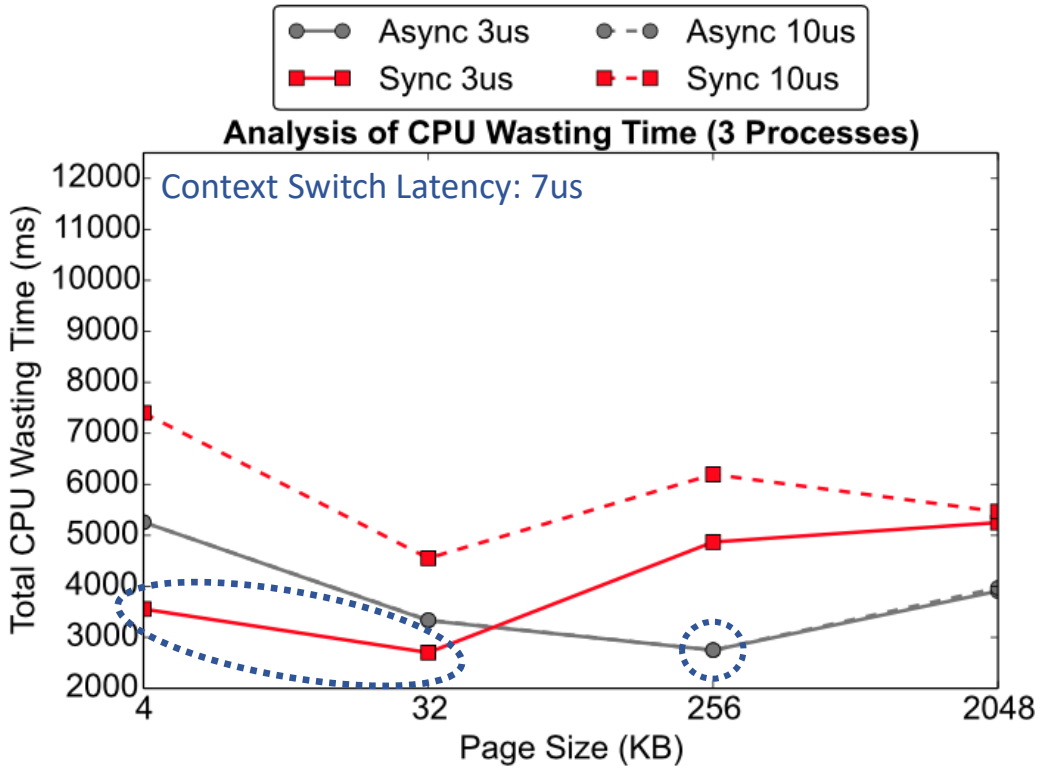


# Hybrid Main Memory (3/4)

- **OS-Controlled Memory Extension**
  - Dynamic OS page management to enjoy the benefit of Ultra-Low-Latency (ULL) storage devices and high bus throughput. *(A rethinking of OS Paging)*

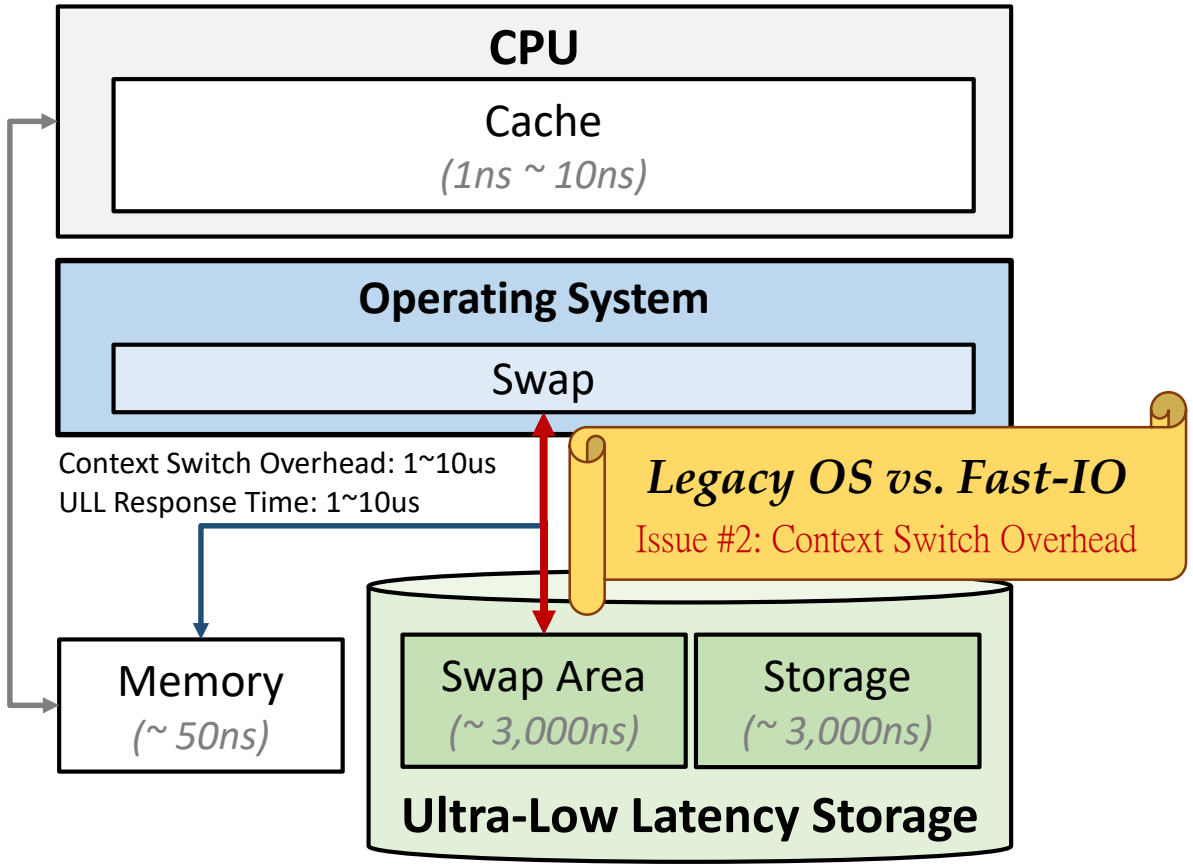


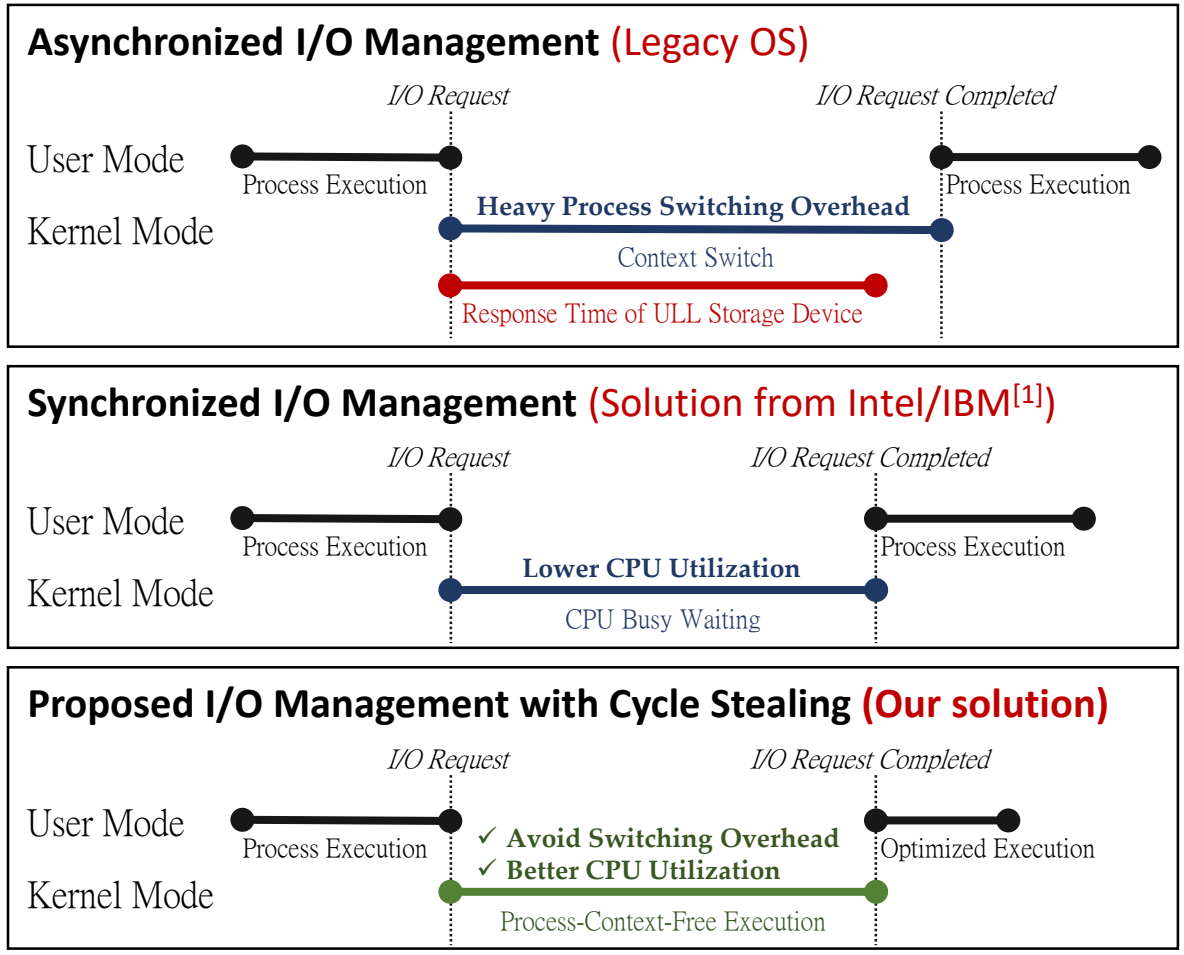




**Larger Page Size Has Better Performance**

# Hybrid Main Memory (4/4)

- **OS-Controlled Memory Extension**
  - Re-design OS management to utilize CPU idle time and enjoy the benefit of Ultra-Low-Latency (ULL) storage devices. *(A rethinking of context switch)*

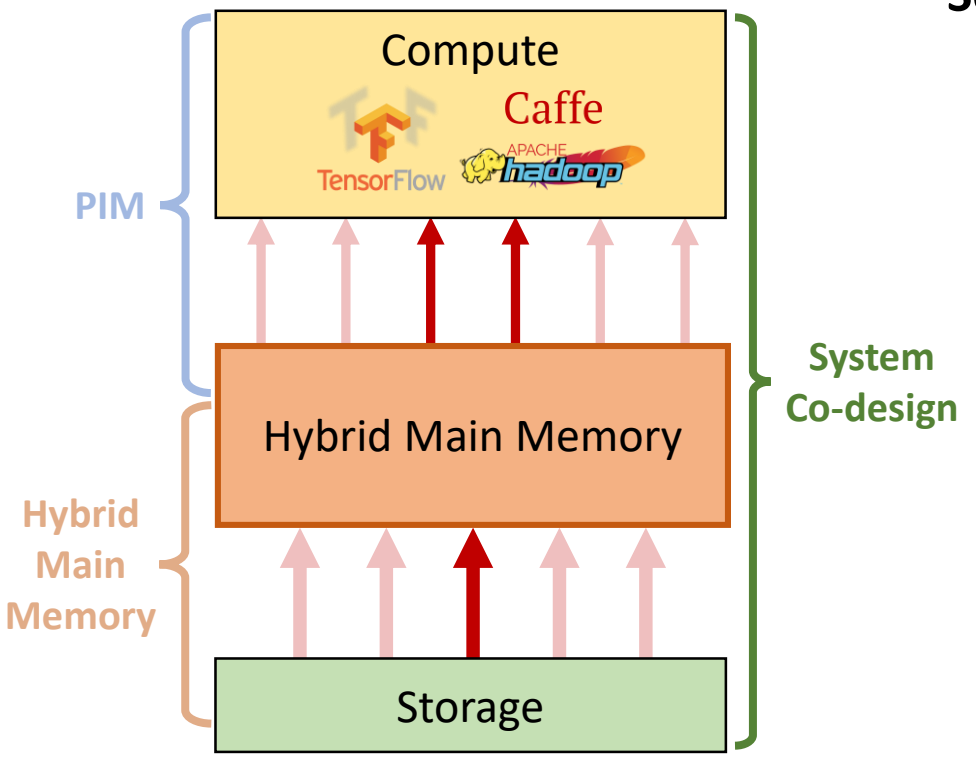



[1] Intel: J. Yang, D. B. Minturn, and F. Hady. When poll is better than interrupt. In Proceedings of the 10th USENIX Conference on File and Storage Technologies, FAST '12, pages 3–3, Berkeley, CA, USA, 2012.

# System Co-Design

- In addition to the unavoidable data movements, device lifetime and system Quality-of-Service (QoS) are also important factors for users.
- Enhance device lifetime and improve system QoS with **Device-Friendly Application Design**.

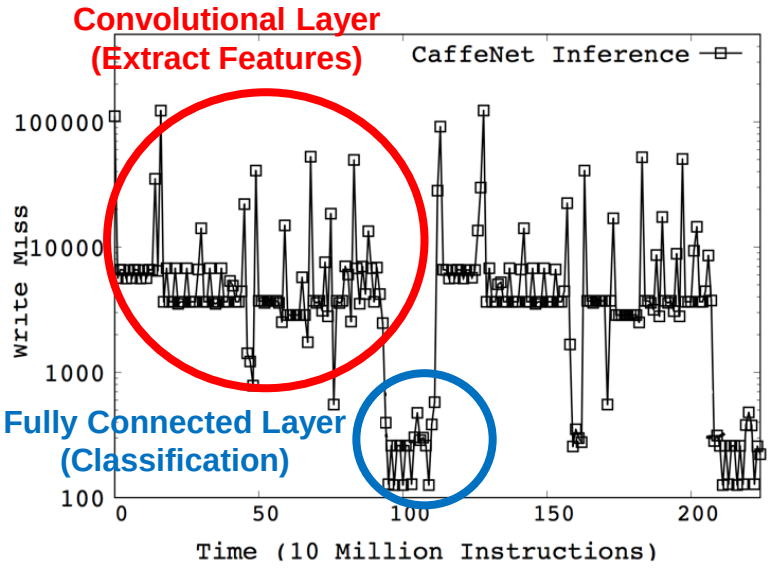


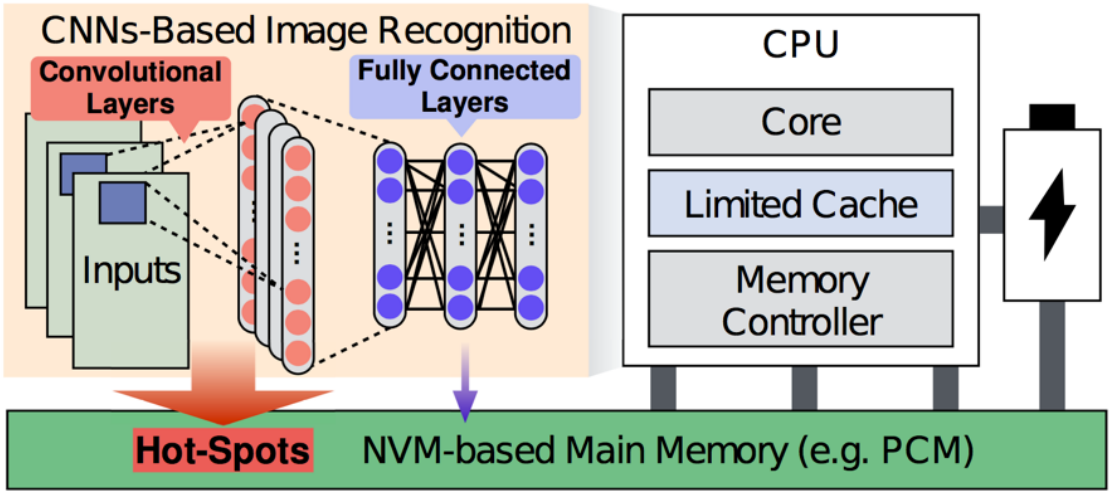

**Some Representative Results**

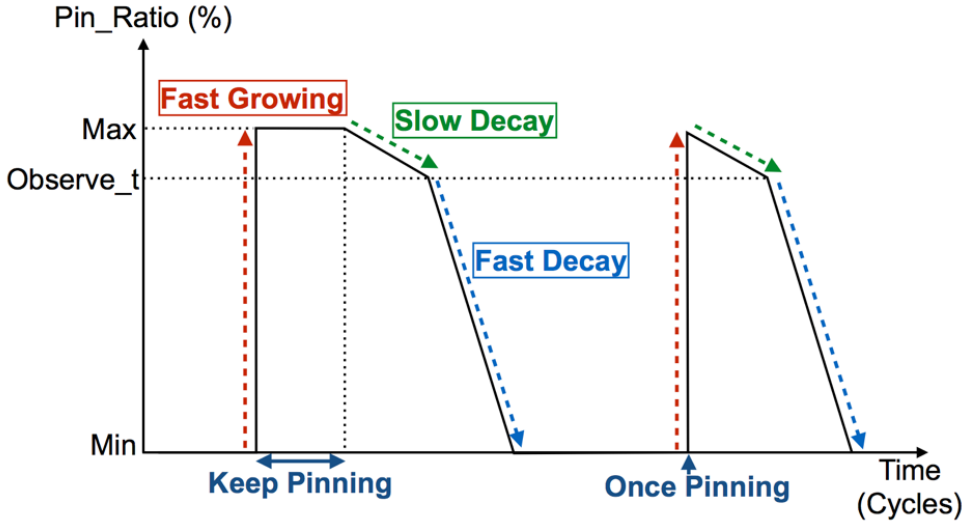
- *Transformer + GPU HBM*
  (NeurIPS' 23, Top Conference)
- *Meta Deep Learning Recommendation System + SSD*
  (ACM/IEEE DAC' 22, Top Conference) (First Author)
- *Convolutional Neural Networks (CNN) + PCM*
  (IEEE TCAD, Integrated with ACM/IEEE EMSOFT' 18, Top Conference) (First Author)
- *Data Deduplication (Cloud Storage) + SMR*
  (IEEE TCAD'21 & ACM/IEEE DAC' 18, Top Conference) (First Author)
- *Random Forests + PCM*
  (ACM SAC' 21, Work with UBO, France)
  (IEEE NVMSA' 19, Work with CUHK, Best Paper Award)
- *Data Consistency + SSD*
  (USENIX OSDI' 20, Top Conference, Work with Academia Sinica)

# Self-Bouncing to Suppress CNN Hot-Spots

- We proposes a *CNN-aware self-bouncing pinning strategy* to efficiently suppress the write hot-spots.
  - Investigate the memory access pattern induced by CNN computing and observe the "write hot-spot" pattern.
  - Propose a CPU cache self-bouncing pinning strategy by leveraging the iterative access pattern of CNN to improve the NVM lifetime.



# An Example of Showing How to Co-Design Applications and Storage Devices

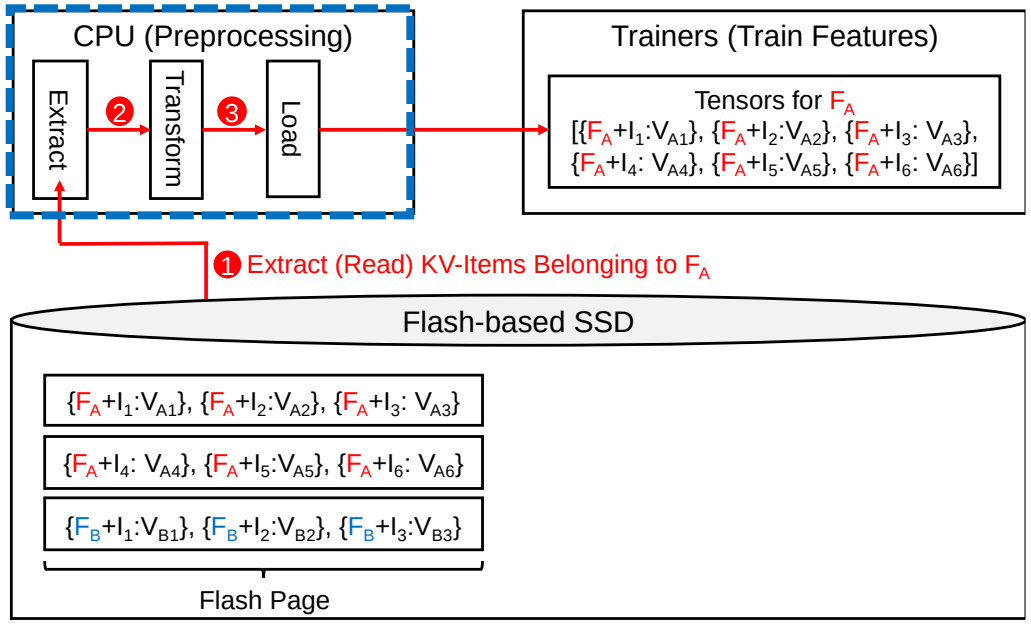
# Co-designing Recommendation Systems with SSDs

- Introduction -- Training Recommendation Systems
- Key Challenge -- Over-Read
- State-of-the-art Solution -- LSM-based Strategy
- Our Solution -- Rec-Aware SSD Data Arranger (READER)
- Evaluation Results

# Training Recommendation Systems

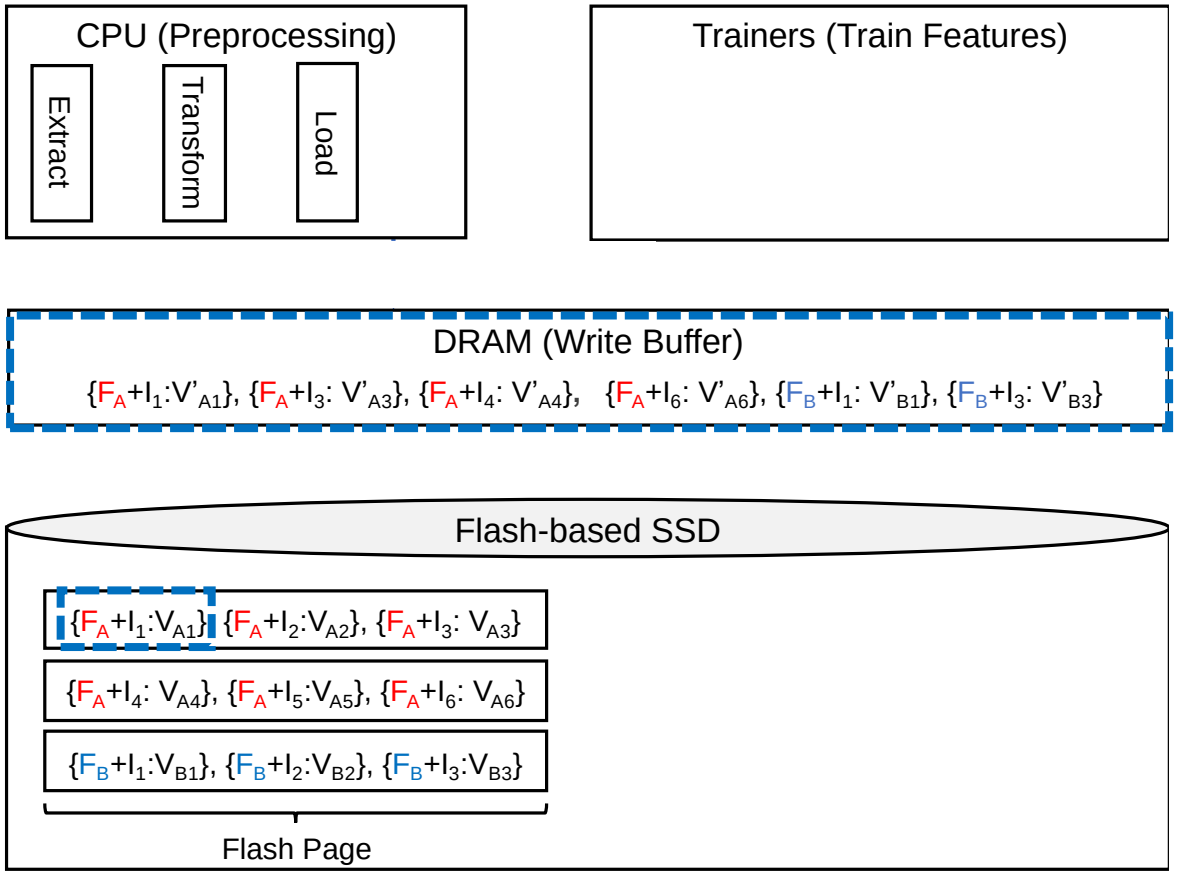
- Recommendation Systems: Widely used in e-commerce and entertainment.
- Feature Training: Data belonging to the same feature will be trained together in trainers.
- Ingest (or preprocess) data for training recommendation systems
  - Extract dataset (raw data) from storage
  - Transform data formats to tensor formats
  - Load to trainers (e.g., GPU or TPU)

| | Movie 1 | Movie 2 | Movie 3 | Movie 4 |
|---|---|---|---|---|
| User 1 | Like | Like | | |
| User 2 | | Like | Like | Dislike |
| User 3 | Like | ? | | Dislike |



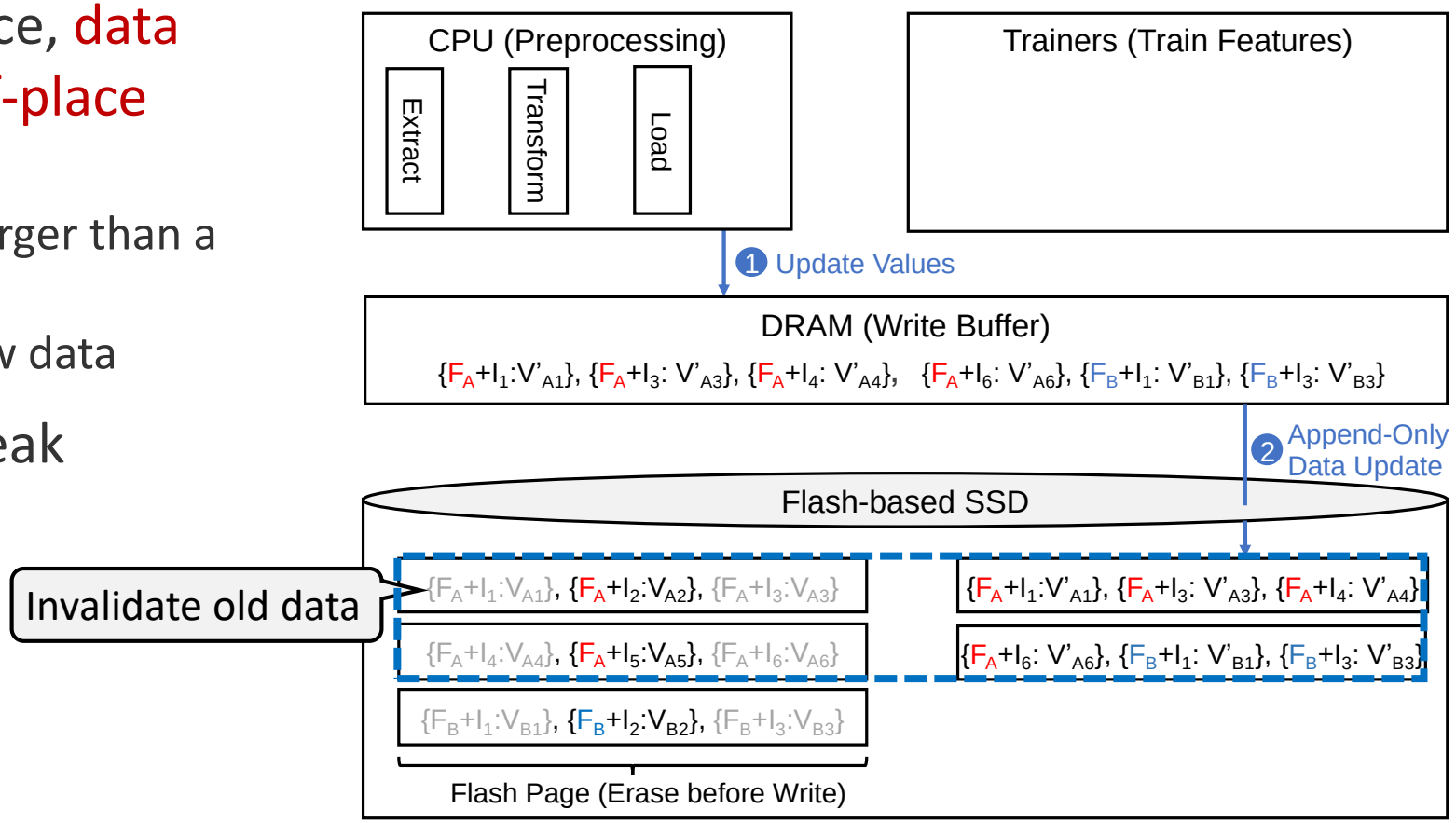
# Update Feature Data in SSD

- Feature is associated with multiple key-value items.
  - Key: Feature + Item (“Rate+Titanic”)
  - Value: Correlation between the feature and the item
- Data is usually not static.
  - Update dataset to the write buffer
  - Or insert new items



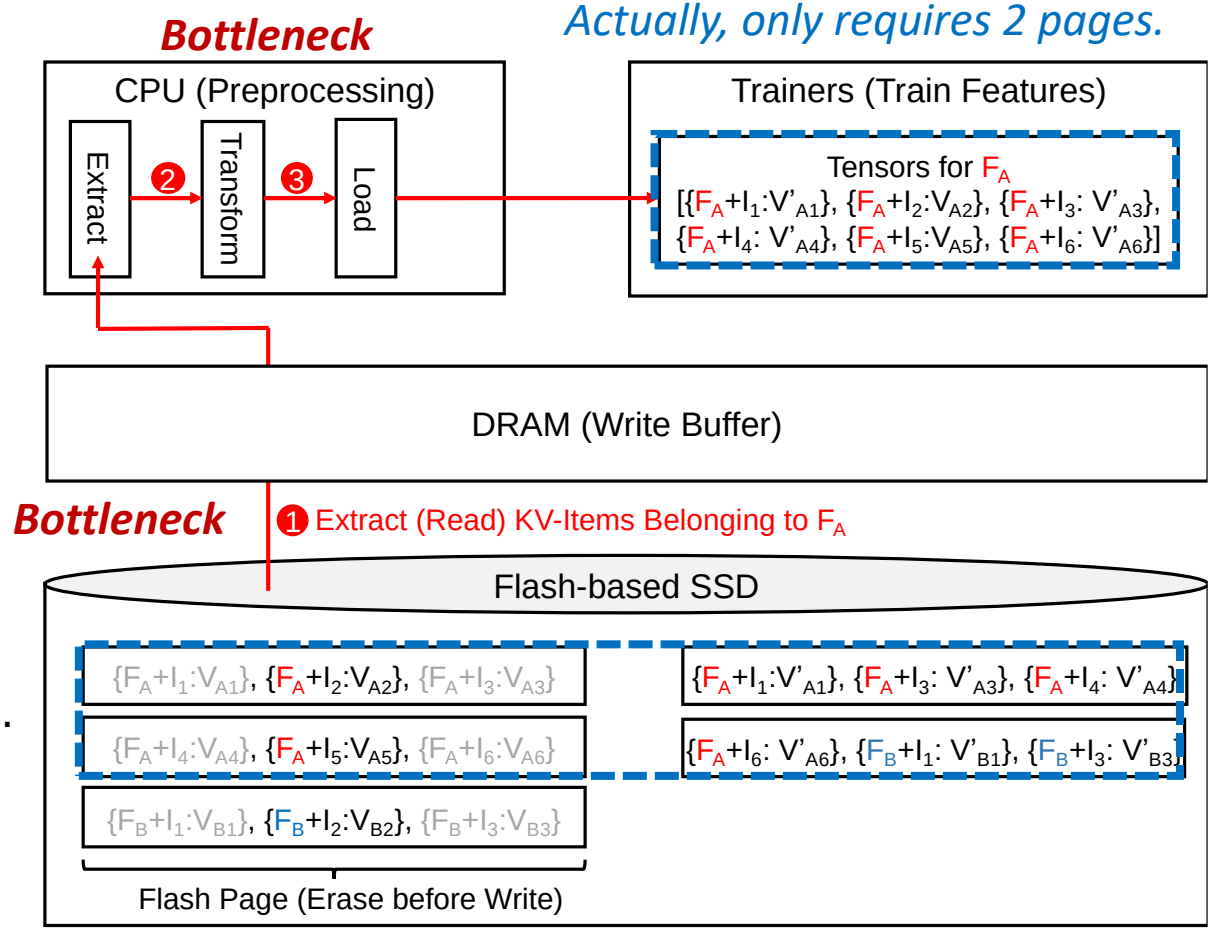
# Update Feature Data in SSD (Cont.)

- Instead of updating data in-place, data in SSDs shall be updated out-of-place
  - SSD's Constraints:
    - An erase unit (i.e., block) is larger than a write unit (i.e., page)
    - Shall erase before writing new data
- Issue: Out-of-place updates break feature locality.



# Challenge of Data Extracting: Over-Read

- Challenge: Most of CPU cycles spent on pre-processing extra data[1].
- Over-Read Issue: CPU reads extra data from SSDs and pre-processes them to reconstruct feature-locality.




Extract $F_A$: Shall read out 4 pages from the SSD.
Extra data: Stale (or Invalid) and unused data.

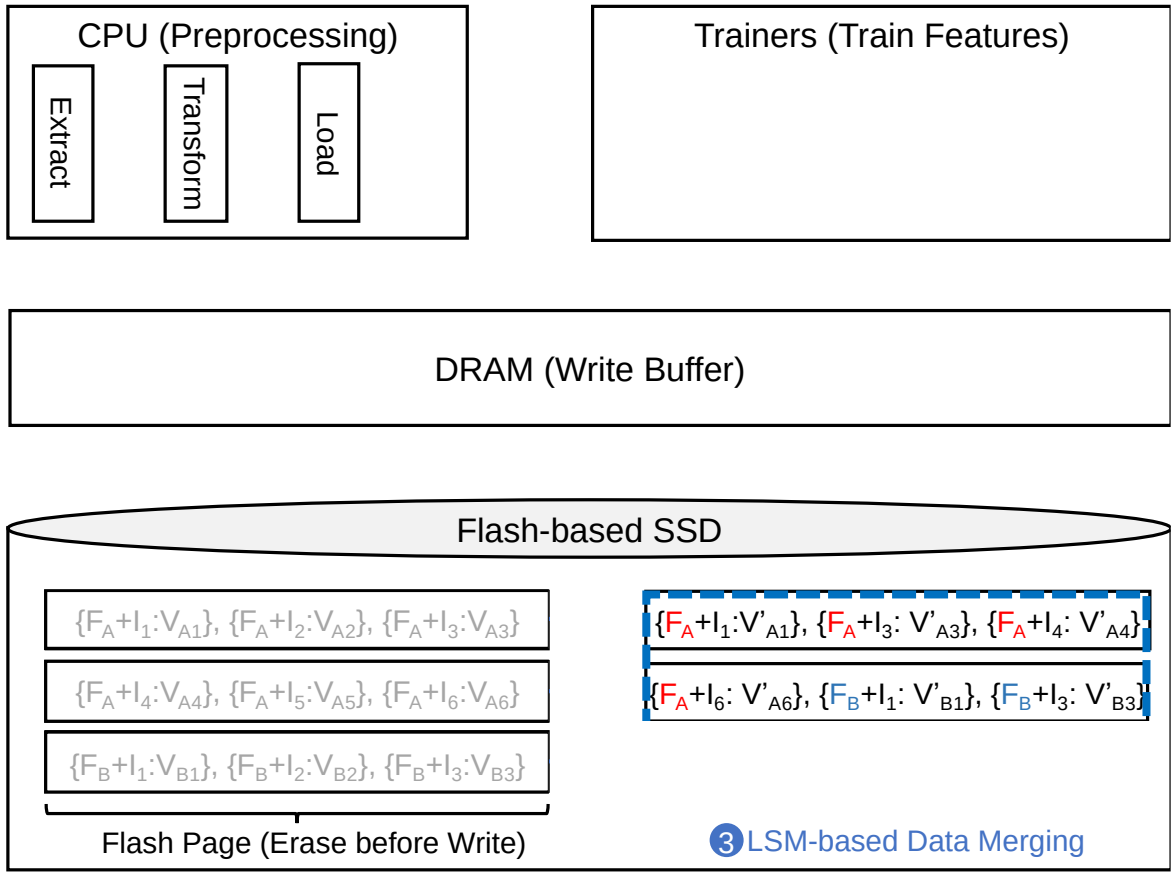
[1] Zhao, Mark, et al. "Understanding and Co-designing the Data Ingestion Pipeline for Industry-Scale RecSys Training." arXiv preprint arXiv:2108.09373 (2021).

# State-of-the-art: Log-Structured Merged (LSM)

- LSM[1]: Merges and writes valid data from low-utilized pages to new pages. (Avoid reading stale data.)
- Challenge: Feature Items related to $F_A$ are still scattered. (Mix with other features)

Extract $F_A$: Read out 3 pages from SSD.

Extra data: ~~Stale (or Invalid)~~-and unused data.

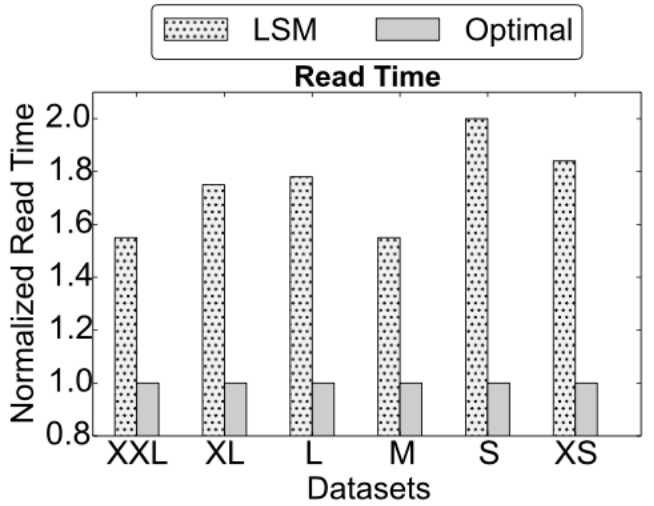


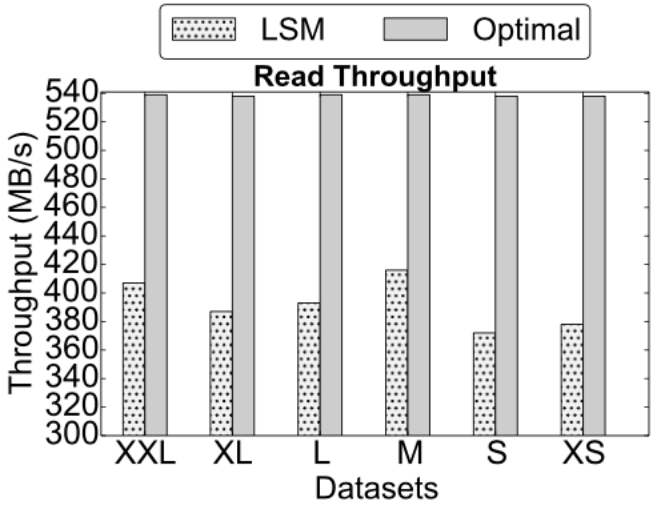

[1] Weijie Zhao, Deping Xie, Ronglai Jia, Yulei Qian, Ruiquan Ding, Mingming Sun, and Ping Li. 2020. Distributed hierarchical gpu parameter server for massive scale deep learning ads systems. arXiv preprint arXiv:2003.05622 (2020)

# Does LSM Effective Enough?

- Latency: LSM spends 1.53x ~ 2x longer than optimal
- Bandwidth: Drops by 22% ~ 31% compared with optimal
- *Challenges: How to have a good SSD's data arrangement so as to avoid reading unused data?*
  - Pro-actively merge data belonging to the same feature.
  - Smartly pin data in the write buffer.

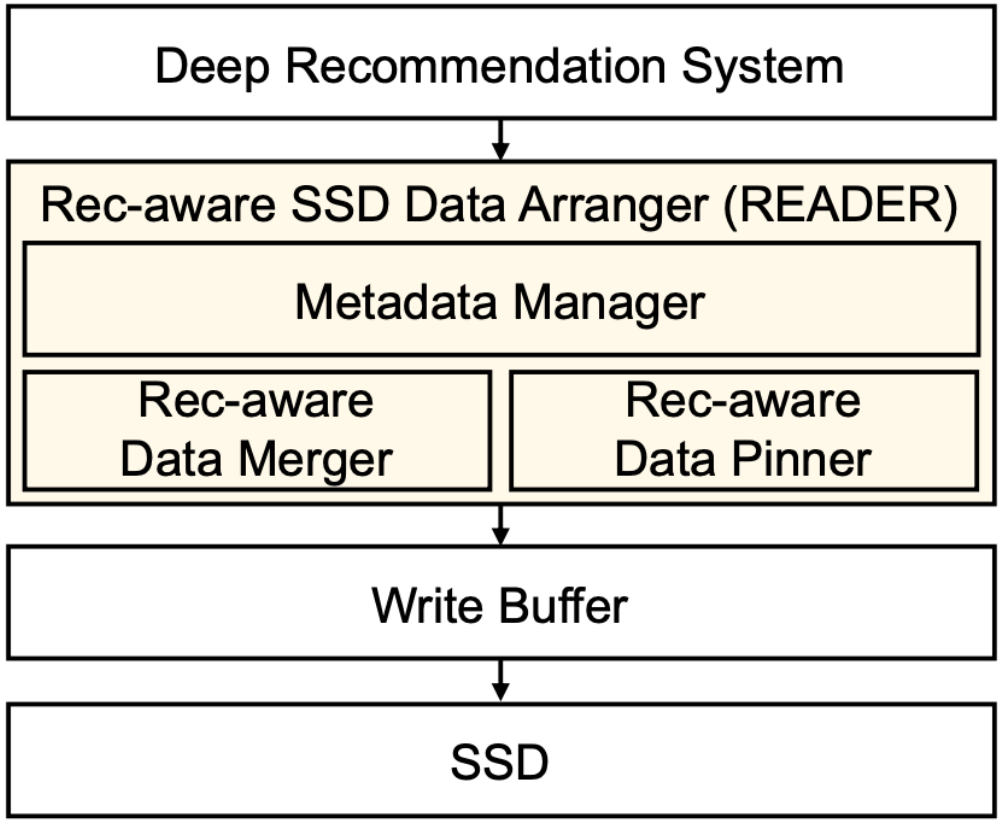
| | XXL | XL | L | M | S | XS |
|---|---|---|---|---|---|---|
| KV items (Billions) | 1.6 | 0.8 | 0.51 | 0.31 | 0.16 | 0.08 |



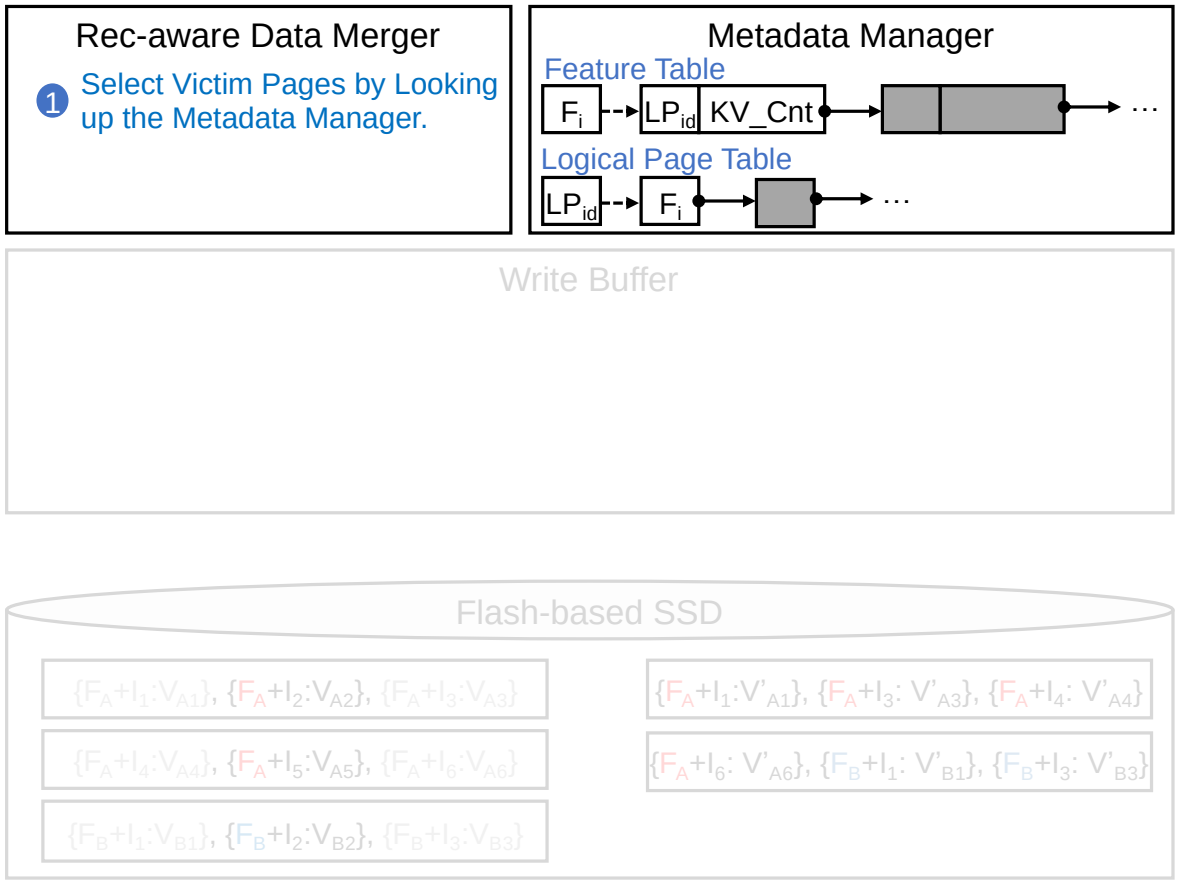
# Rec-Aware SSD Data Arranger (READER)

- Our READER comprises:
  - Rec-aware Data Merger: Merge (rearrange) the scattered feature items to new pages.
  - Rec-aware Data Pinner: Select and pin data inside buffer with considering feature-locality.



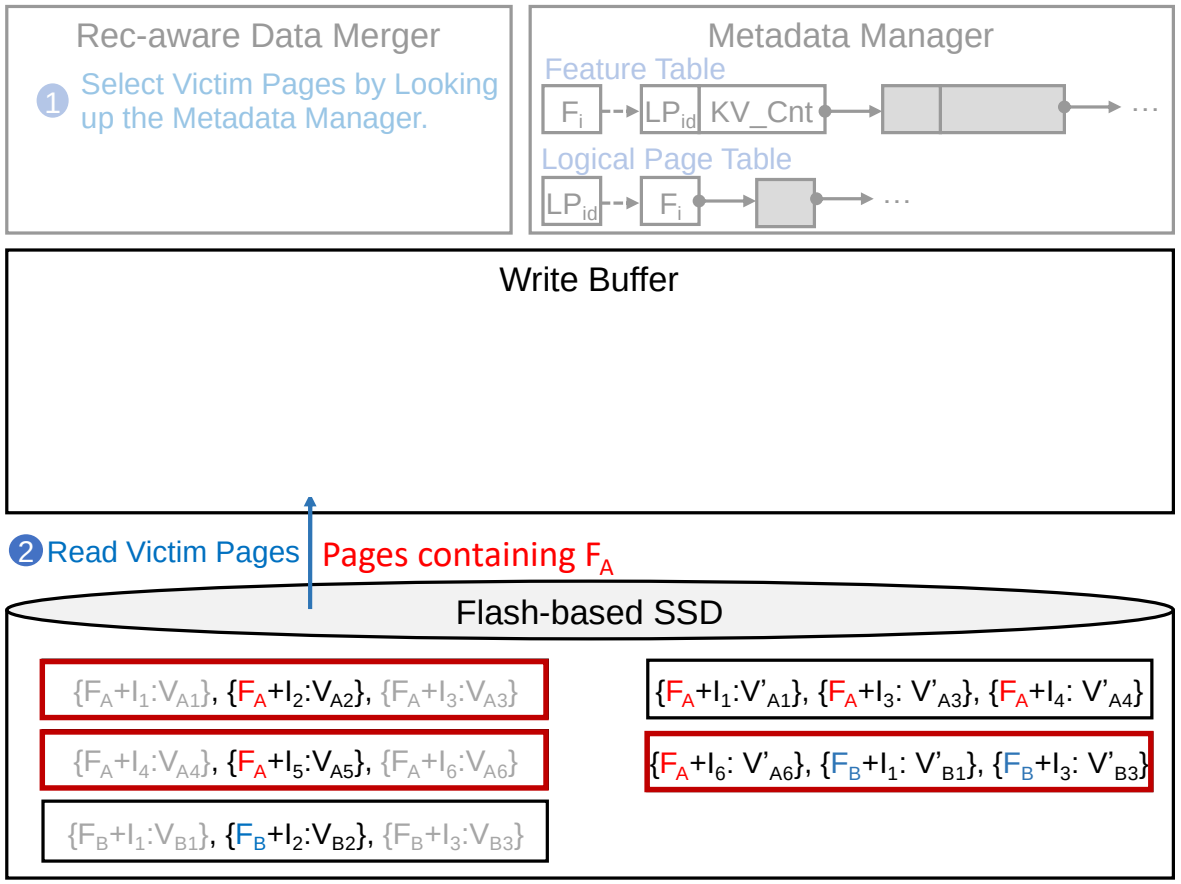
# Rec-aware Data Merger

- Select the to-be-merged features and merges the scattered feature items to new pages
- Metadata Manager:
  - Feature Table:
    - Mapping between a feature and logical pages.
    - Monitor the number of KV items in each page.
- Feature-fragmentation degree:
  (Current pages store $F_i$)/(Ideal pages store $F_i$)



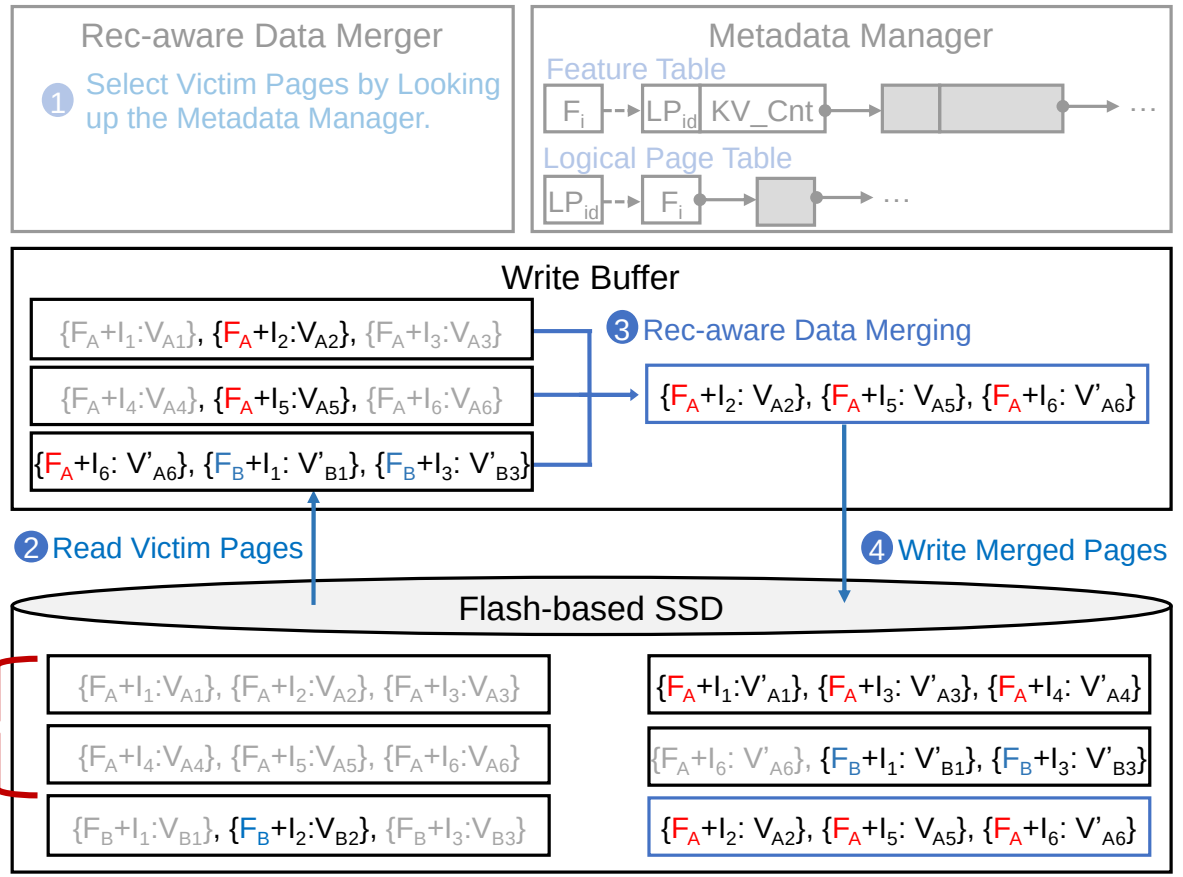
# Rec-aware Data Merger (Cont.)

- Select the to-be-merged features and merges the scattered feature items to new pages
  - 2. Read victim pages
  - 3. Merges data items belonging to the same feature



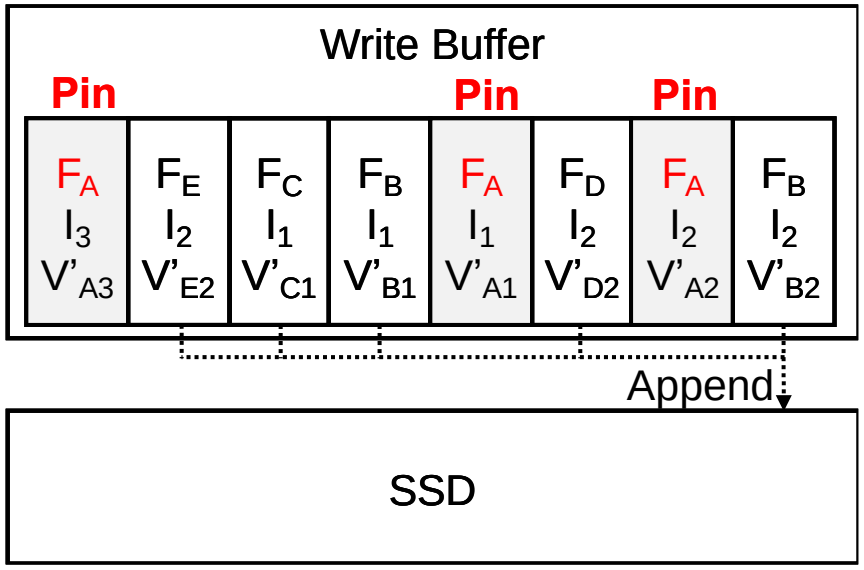
# Rec-aware Data Merger (Cont.)

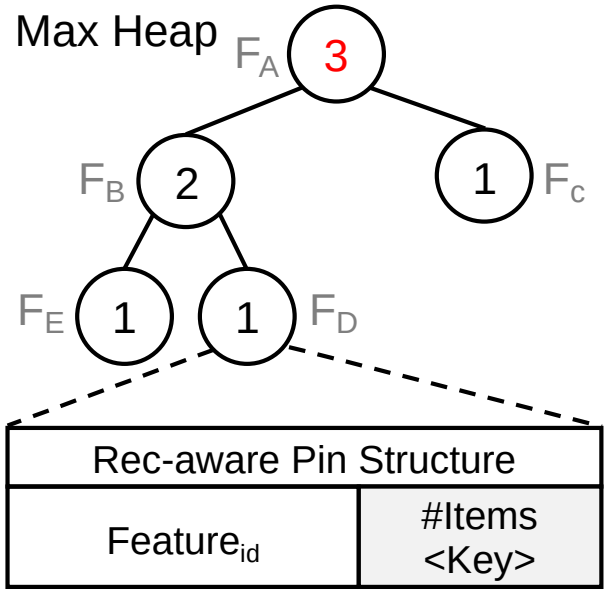
- Select the to-be-merged features and merges the scattered feature items to new pages
  - 4. Write merged pages to SSD
  - 5. Update metadata manager
  - Do garbage collection in background



# Rec-aware Data Pinner

- Challenge: KV-items belonging to different features may be gradually mixed together until the data merger merges the feature again.
- Main Idea: Increase feature locality by pinning items belonging to the same feature in the write buffer.



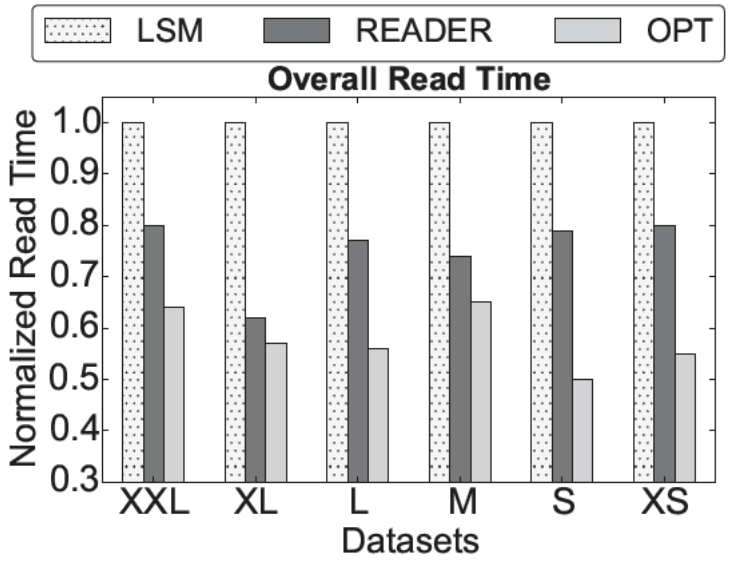



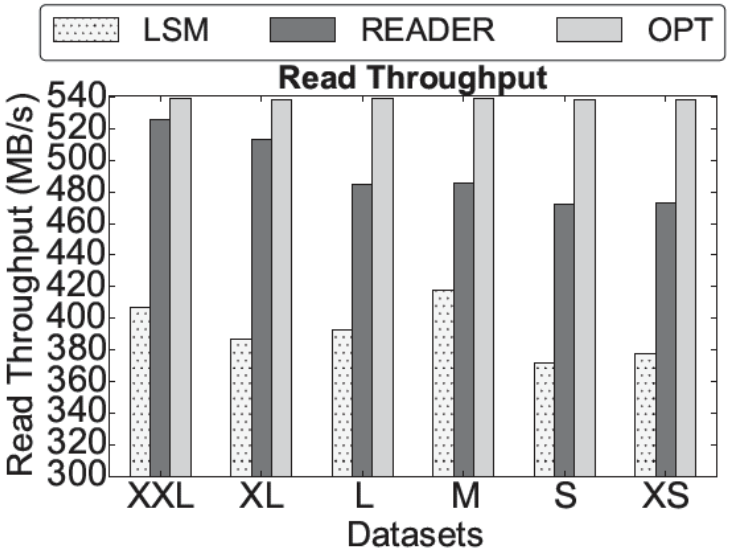
# Evaluation: Read Performance

- Read Time: READER can save 20% ~ 38% of overall read time compared with the LSM-based strategy.
- Read Throughput: READER can reach 1.16X ~ 1.33X higher read throughput than the LSM-based strategy.
  - READER's throughput results are within 12% of the optimal read strategy.

| | XXL | XL | L | M | S | XS |
|---|---|---|---|---|---|---|
| KV items (Billions) | 1.6 | 0.8 | 0.51 | 0.31 | 0.16 | 0.08 |
| Features | 16000 | 8000 | 6400 | 5200 | 4000 | 800 |

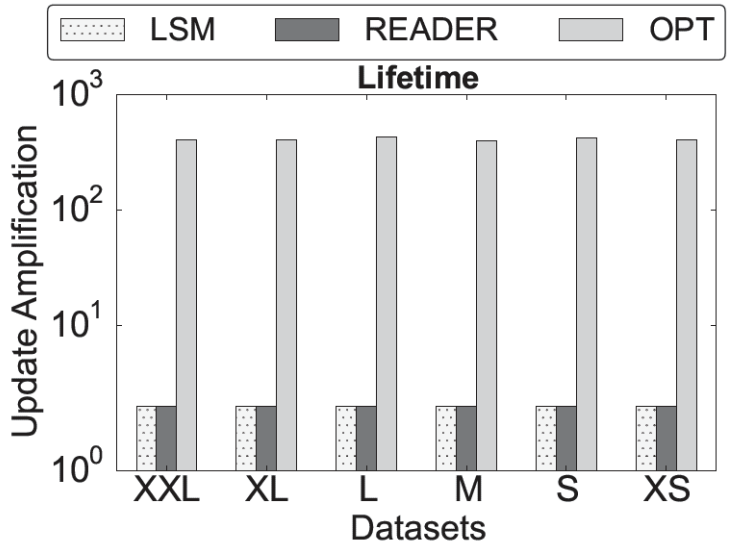
Datasets from Kaggle

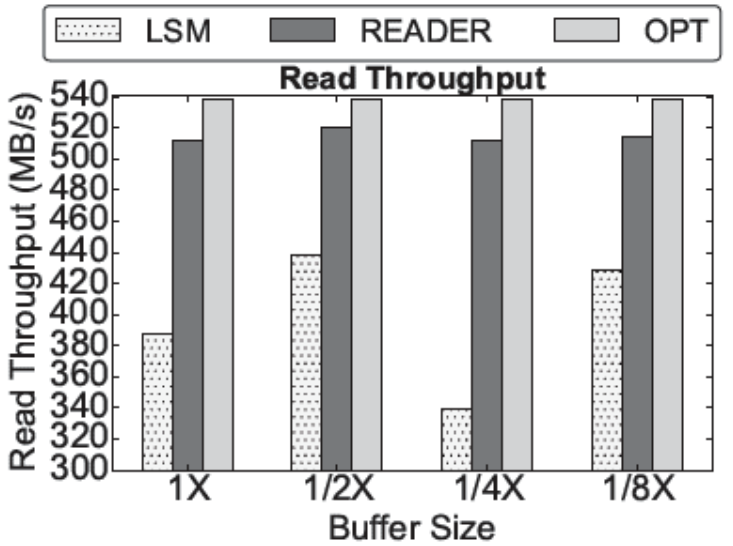


# Evaluation: Lifetime & Buffer Size

- Lifetime (Update Amplification): The optimal read strategy requires around 400 extra writes for updating an item. READER and the LSM-based strategy only need 1 extra writes for updating each item.
- Smaller Write Buffer: READER can reach 1.19X ~ 1.51X higher read throughput than the LSM-based strategy.
  - Also, READER is more stable



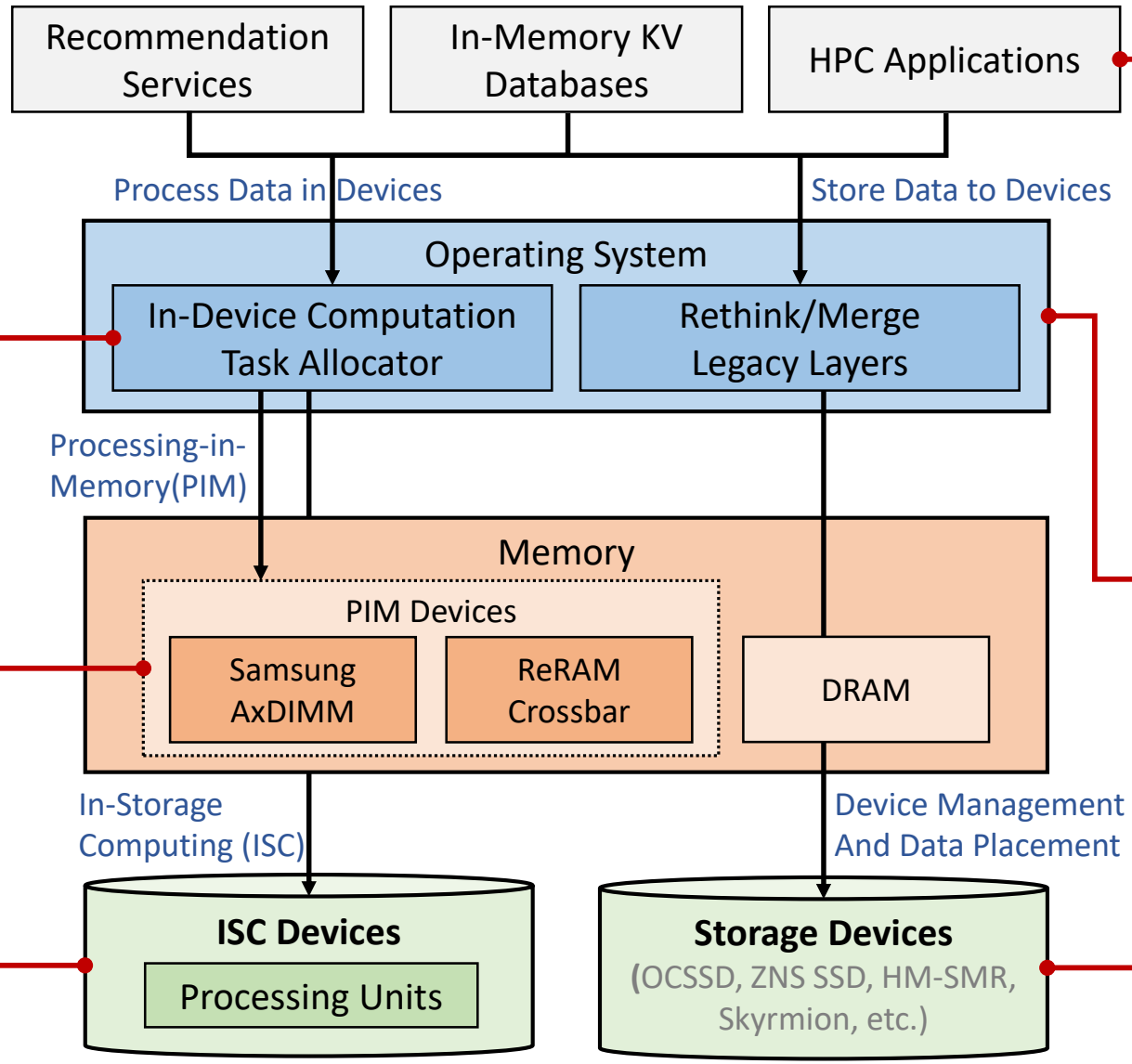
# Summary

- Reading unused feature items from SSDs seriously hurts the performance of recommendation model training.
- We present a joint management middleware, a rec-aware SSD data arranger (READER), to rearrange data in SSDs with considering the behavior of recommendation model training.
  - Rec-aware data merger: pro-actively merges items belonging to the same feature.
  - Rec-aware data pinner: smartly pins items in the write buffer.
- READER can save the overall read time by 20% ~ 38% compared with the LSM-based strategy, and our read throughput results are within 12% of the optimal case.

# What are we doing now?

*Computing-Centric*

*Data I/O-Centric*




**Joint Designs**

1. Harmonize PIM and ISC devices.
2. Integrate legacy and emerging computing units

**Processing-in-Memory (PIM)**

1. AxDIMM or UPMEM
2. Memristor-based Crossbar
3. Photonic-based PIM

**In-Storage Computing (ISC)**

1. Design and Profile In-SSD Computation
2. Operating System inside ISC Devices

**Application Exploration**

1. Graph Neural Networks
2. Recommender Systems
3. Large Language Models
4. In-Memory KV DBs

**OS Rethinking**

1. Rethink Memory Layers
2. In-Memory File System
3. Intermittent Computing

**Device Exploration**

1. White-Box (e.g., OCSSD, SMR)
2. Compute-Express Link (CXL)
3. Emerging Devices (Photonic, Skyrmion, DNA etc.)

*Thanks for your attention*

*Q & A*