DALdex: A DPU-Accelerated Persistent Learned Index via Incremental Learning

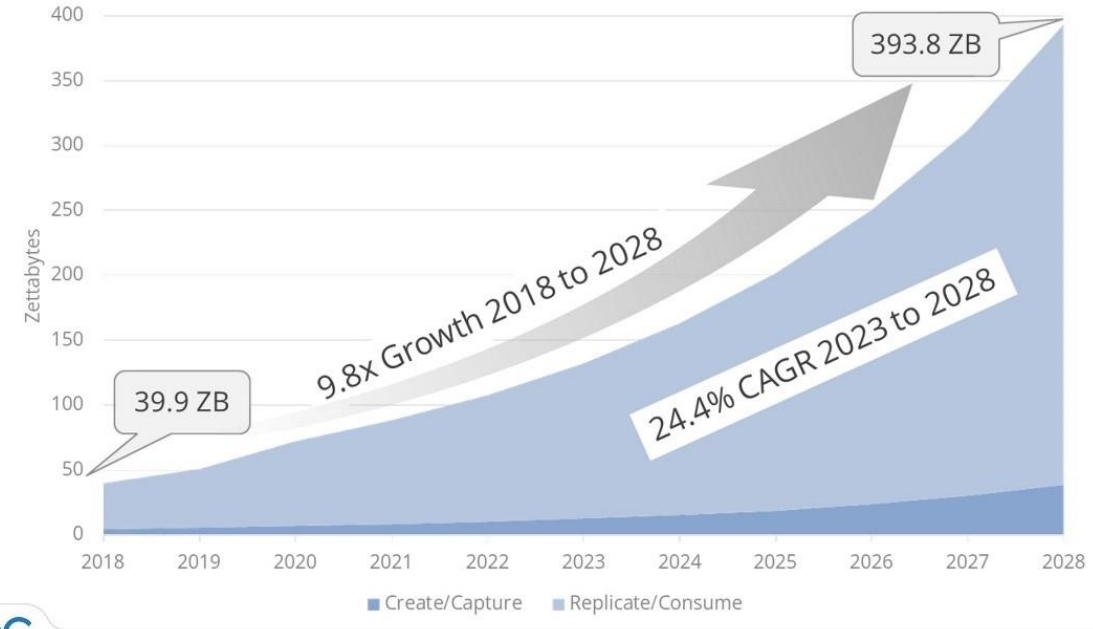
Aoyang Tong, Yu Hua, Menglei Chen

Huazhong University of Science and Technology

39th ACM International Conference on Supercomputing (**ICS**), 2025

Bottleneck: In-Memory Index

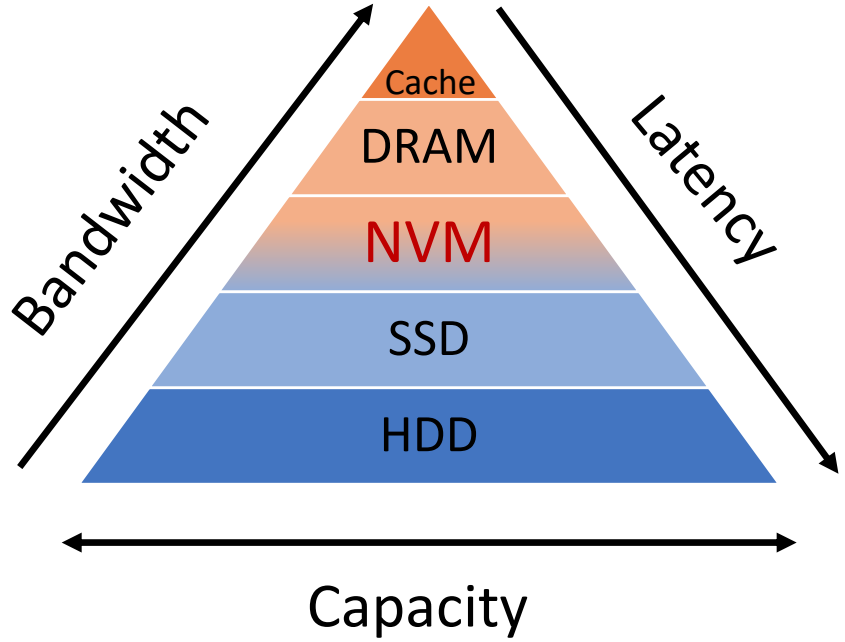
- Traditional ***B/B⁺Tree*** are widely employed in HPC systems.
  - ***Memory bottleneck***: limited scalability in capacity.
  - ***Index bottleneck***: unaware of data distribution patterns.



[1] Source: https://my.idc.com/getdoc.jsp?containerId=prCHC52667624

Solution: HW & SW Co-Design

- HW solution: ***Non-Volatile Memory*** (e.g., Intel Optane DC PMEM, CXL-SSD)
  - DRAM-like Byte-addressability & Storage-like Capacity.

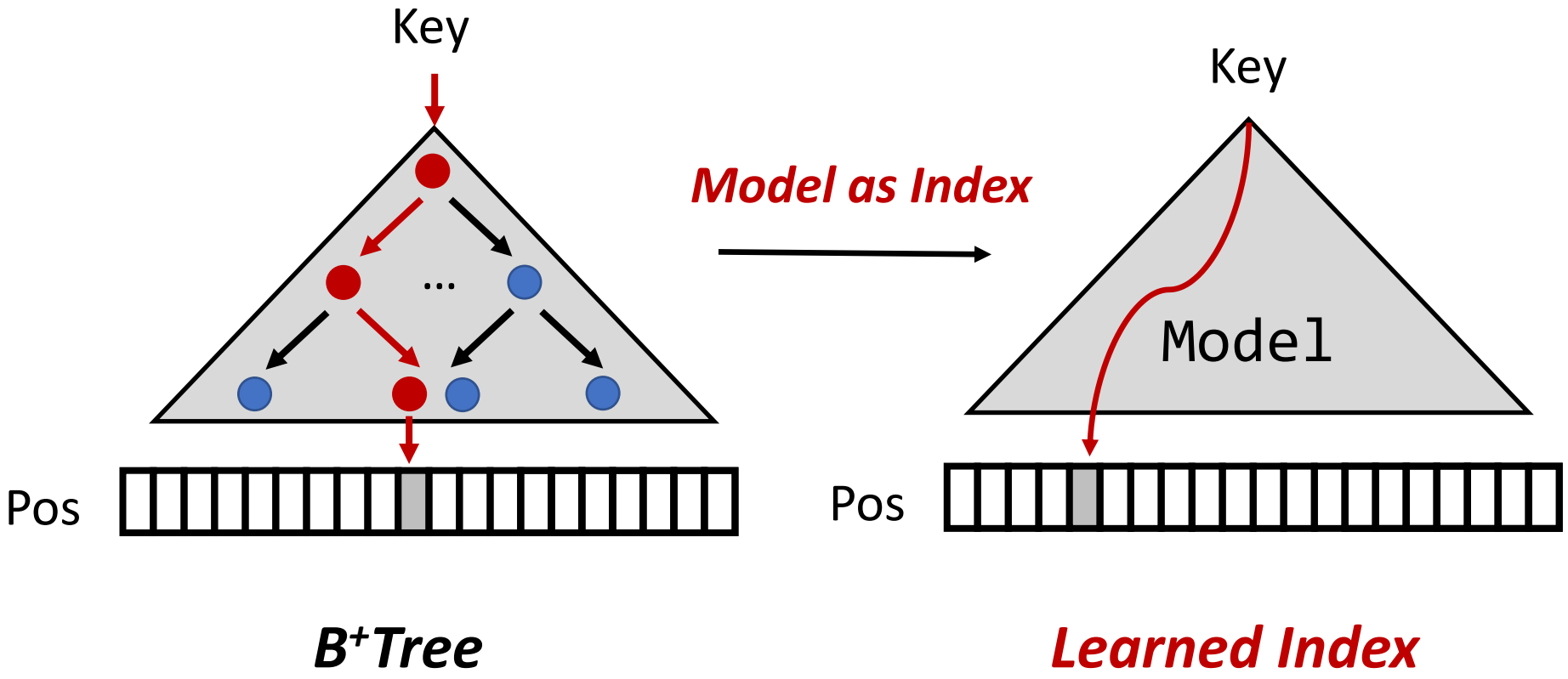


Intel Optane DC PMEM[1]

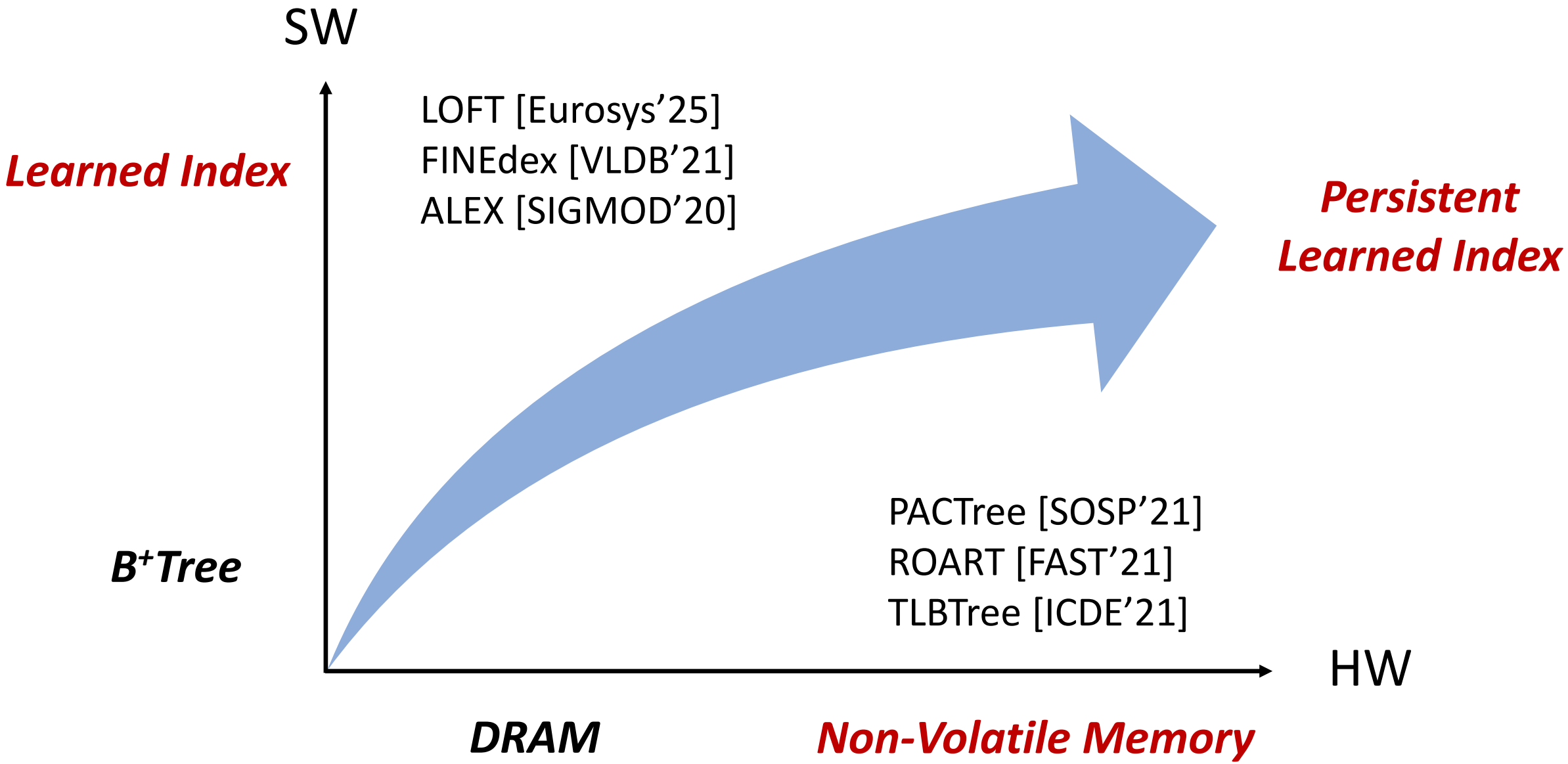
[1] Source: https://www.intel.cn/content/www/cn/zh/architecture-and-technology/optane-dc-persistent-memory.html

Solution: HW & SW Co-Design

- HW solution: ***Non-Volatile Memory*** (e.g., Intel Optane DC PMEM, CXL-SSD)
  - DRAM-like Byte-addressability & Storage-like Capacity.
- SW solution: ***Learned Index*** (e.g., neural network, linear regression).

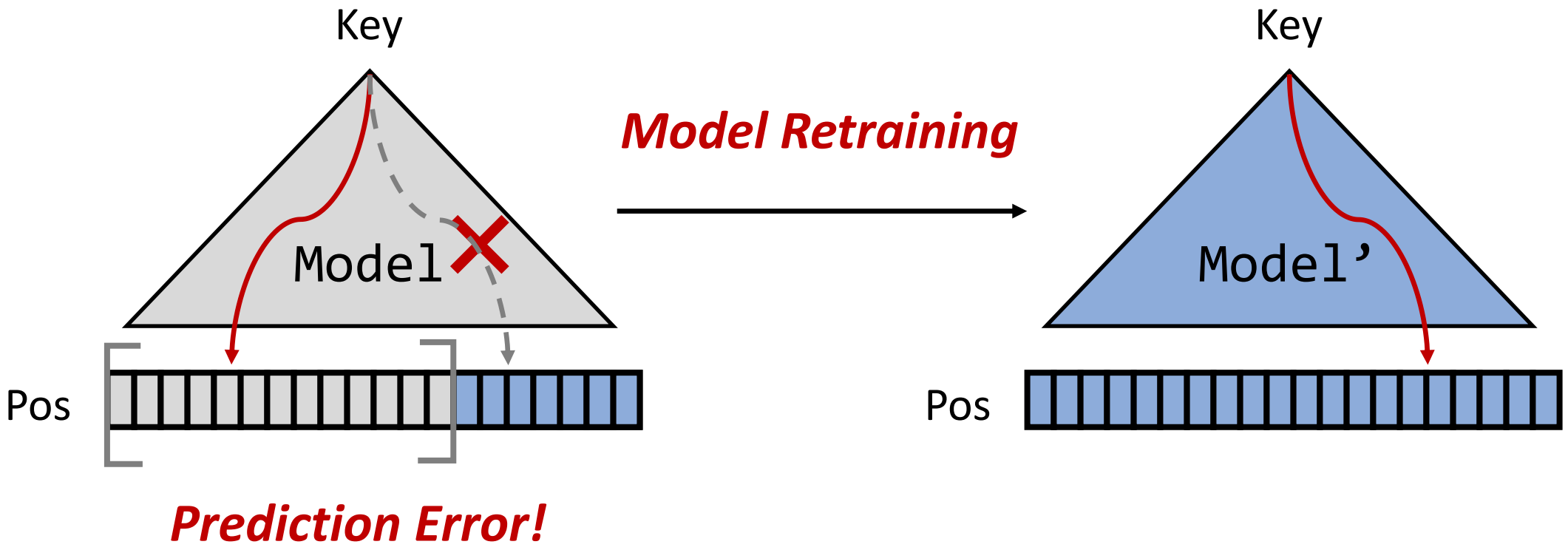


Solution: HW & SW Co-Design



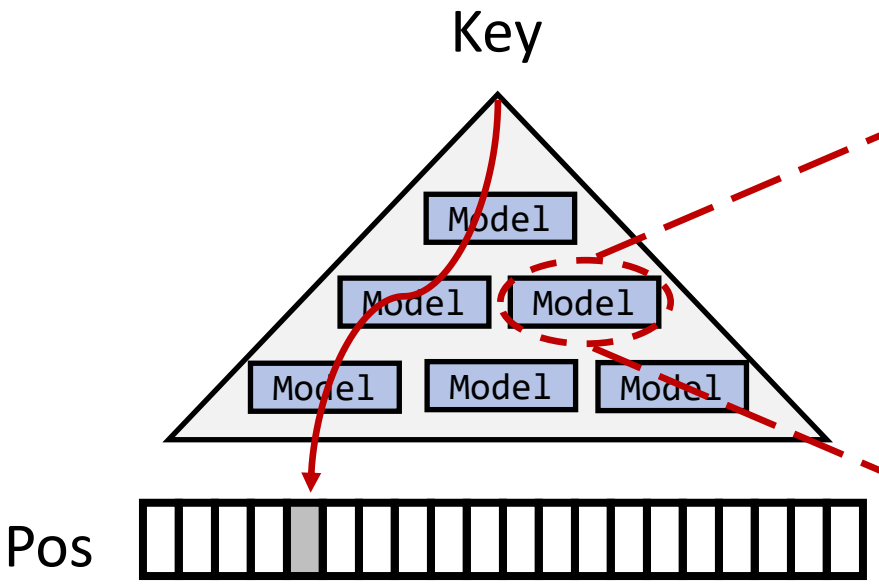
Challenge #1: Expensive Model Retraining

- Learned indexes require frequent ***model retraining*** to learn new data distributions.



Challenge #1: Expensive Model Retraining

- Learned indexes require frequent ***model retraining*** to learn new data distributions.
  - Model retraining is ***time-consuming***, especially on low-speed storage devices.



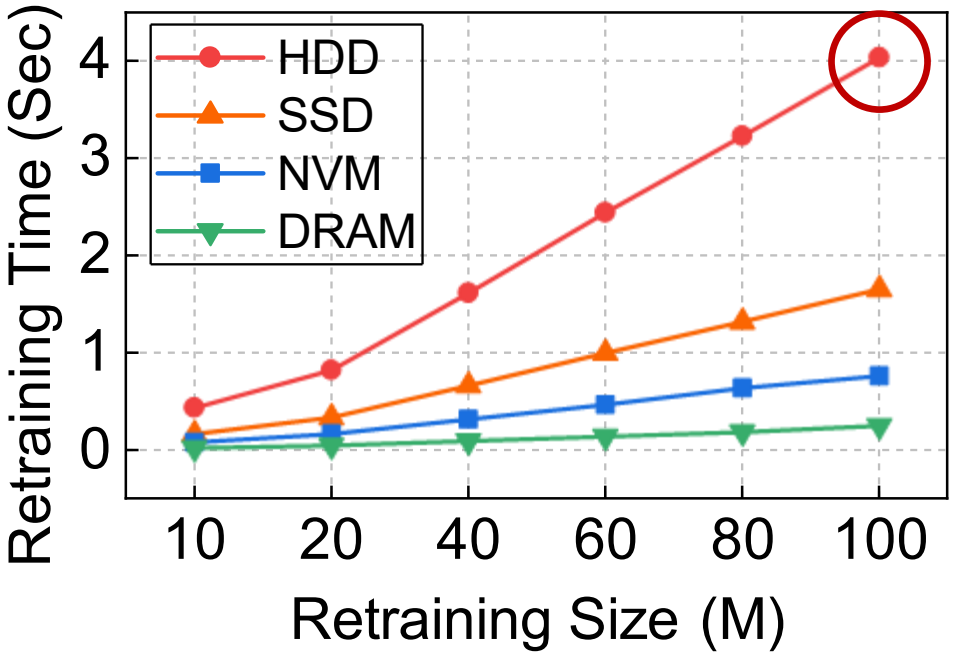
$$y = ax + b^{[1]}$$

$$a = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$b = \bar{y} - a\bar{x}$$

Bottleneck!

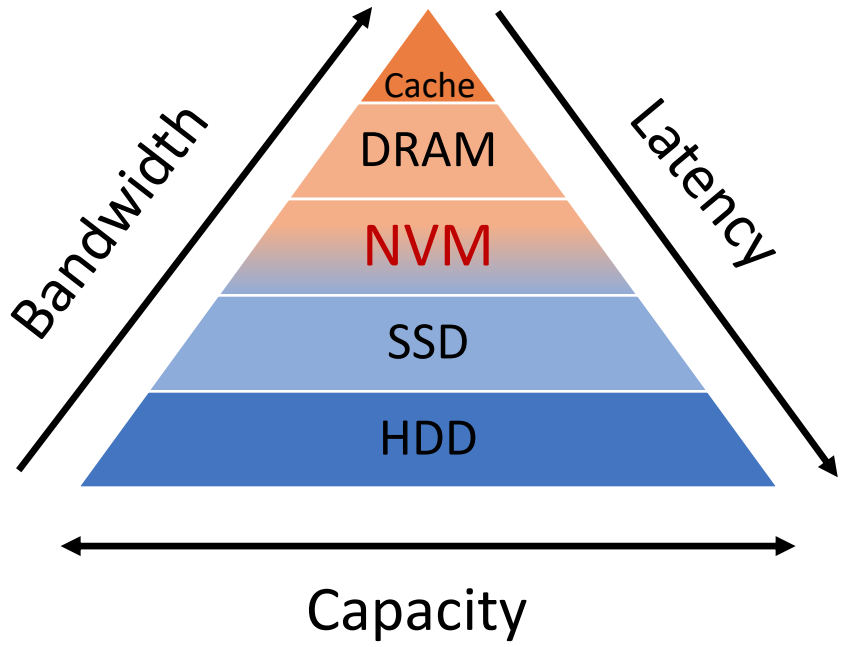
4 Seconds!



[1] One-dimensional linear regression using least squares method.

Challenge #2: Excessive NVM Access

- Model access is on the ***critical path*** of persistent learned indexes.
  - NVM exhibits ***lower*** performance metrics than DRAM.

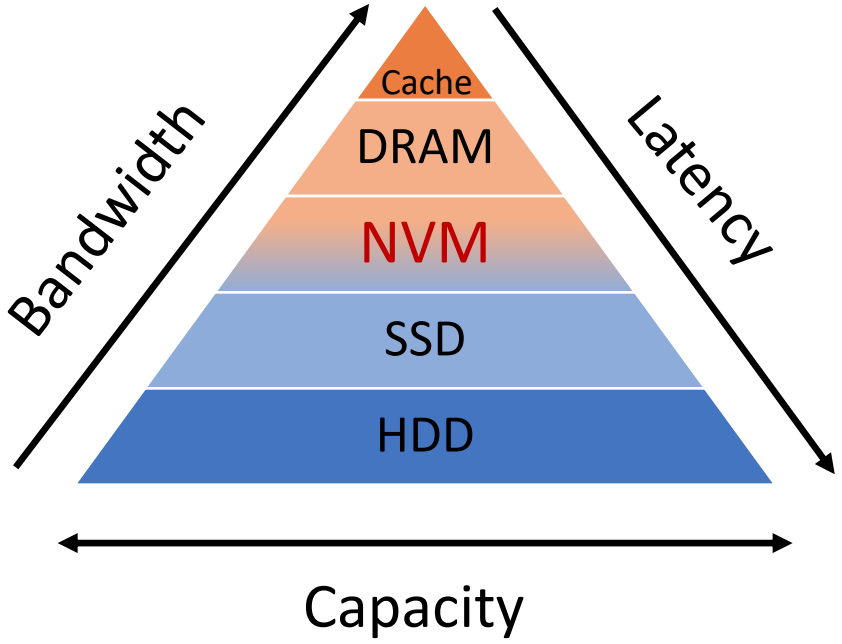


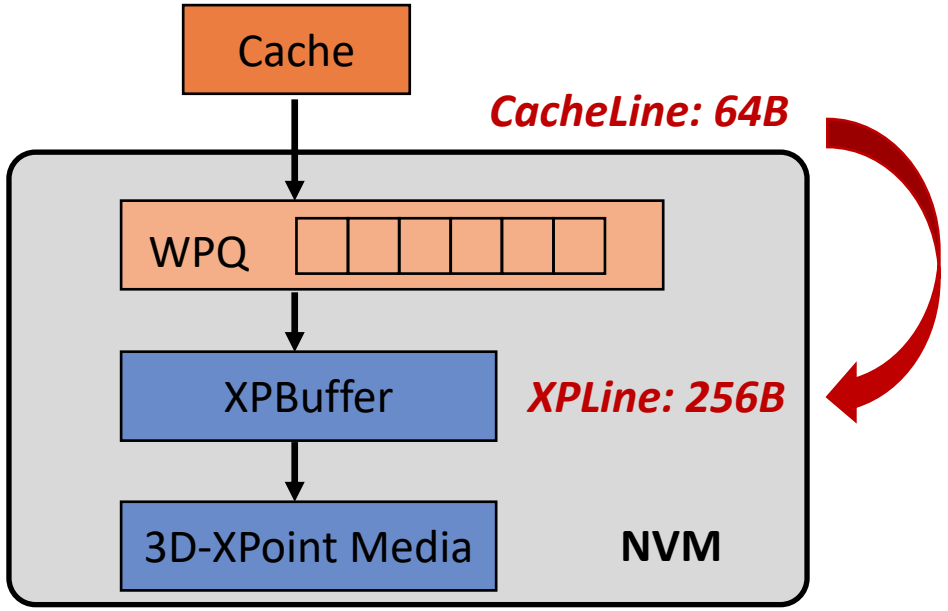
Per DIMM	Optane DC PMEM	DRAM
Latency	~170ns	~75ns
Read Bandwidth	~7.6 GB/s	~15 GB/s
Write Bandwidth	~2.3 GB/s	~15 GB/s

Limited Bandwidth!

Challenge #2: Excessive NVM Access

- Model access is on the ***critical path*** of persistent learned indexes.
  - NVM exhibits ***lower*** performance metrics than DRAM.
  - Mismatched access granularity between CPU Cache and NVM.

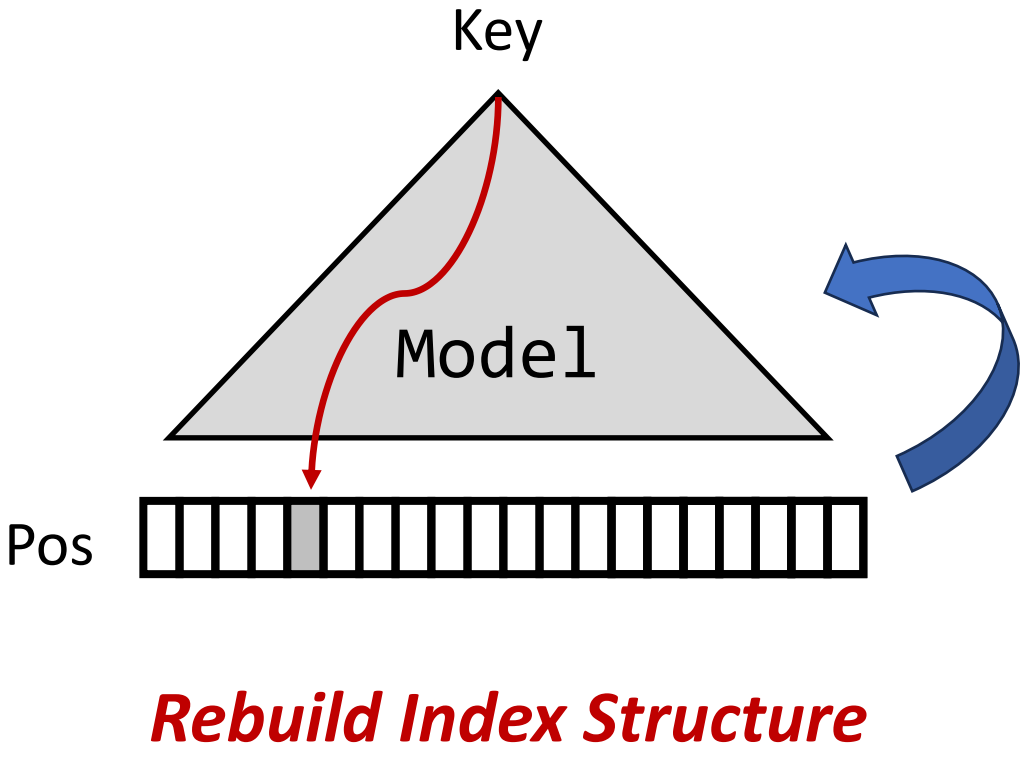


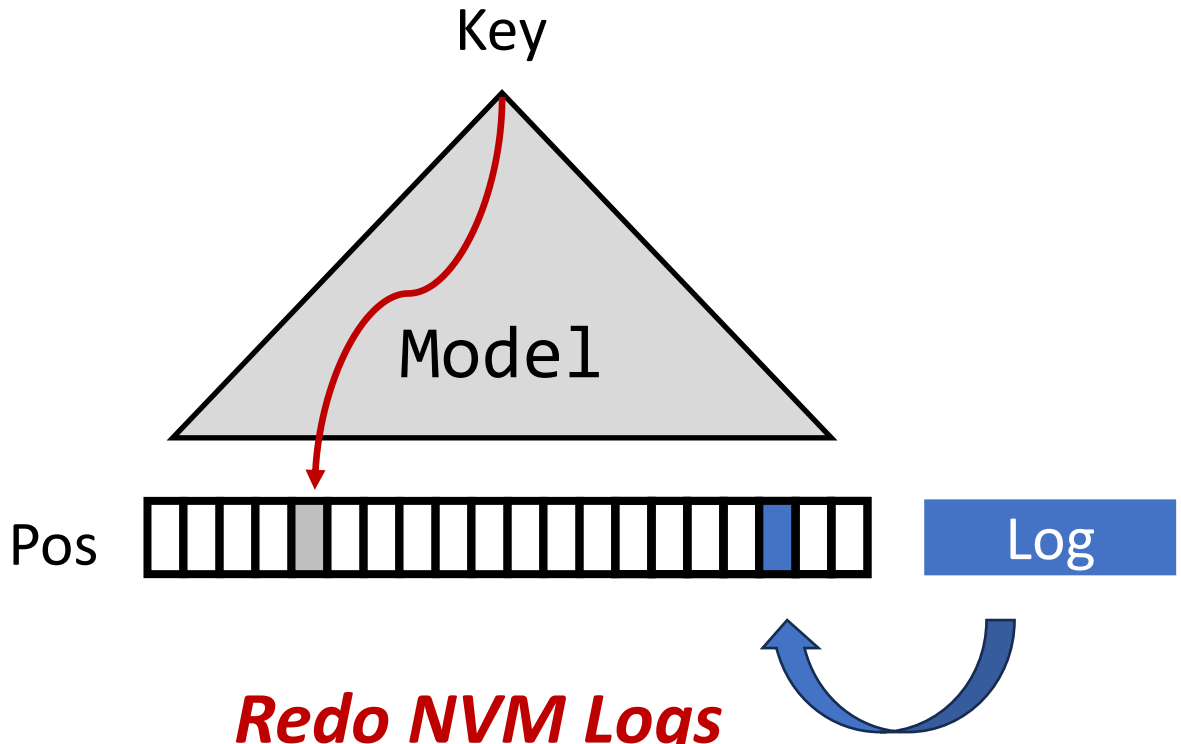


4X Amplification!

Challenge #3: Inefficient Recovery Mechanism

- State-of-the-art persistent learned indexes have to ***rebuild partial index structures***[1] or ***redo heavy NVM logs***[2] during recovery.

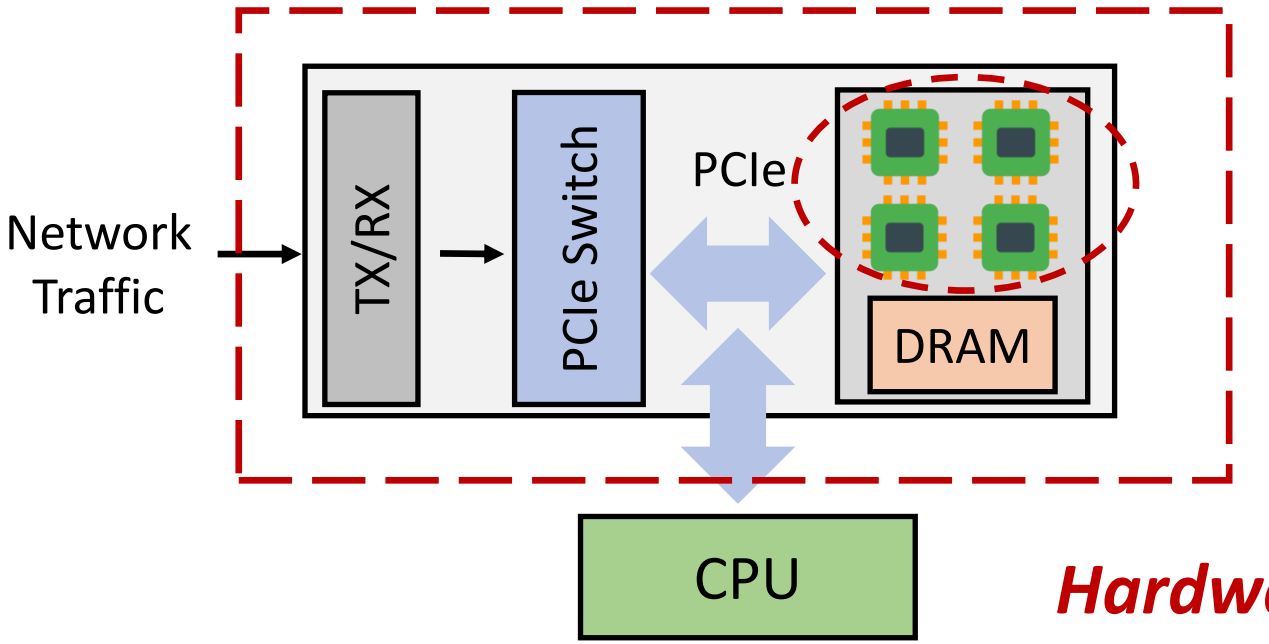




[1] APEX: A High-Performance Learned Index on Persistent Memory, *VLDB'2022*
[2] PLIN: A Persistent Learned Index for Non-Volatile Memory with High Performance and Instant Recovery, *VLDB'2022*

DALdex: DPU-Accelerated Learned Index

- NVIDIA BlueField Data Processing Unit (DPU).
  - ***Hardware Accelerator***: offload *CPU-intensive* tasks.
  - ***Fault Tolerance***: hardware level isolation[1].



Component	Configuration
Cores	8× ARMv8 A72 processors
DRAM	16GB on-board DDR4
Network	1× Ethernet port 200Gb/s
PCIe	16× PCIe Gen 4.0

Hardware Level Isolation

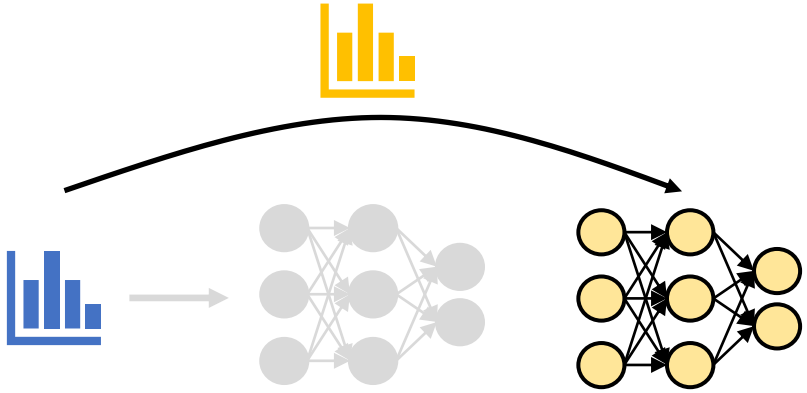
[1] LineFS: Efficient SmartNIC Offload of a Distributed File System with Pipeline Parallelism, *SOSP'21*

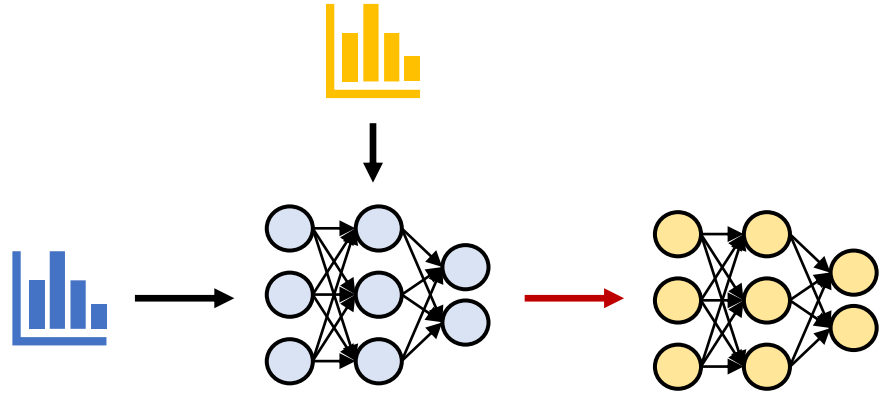
DALdex Design

- Challenge #1: Expensive Model Retraining
  - ***DPU-Offloaded Incremental Learning***
- Challenge #2: Excessive NVM Access
  - ***NVM-friendly Index Structure***
- Challenge #3: Inefficient Recovery Mechanism
  - ***DPU-Assisted Instant Recovery***

Online Incremental Learning

- **Offline Batched Learning**: *train new models from scratch*.
- **Online Incremental Learning**: *retrain old models based on new data distributions.*





Offline Batched Learning ***Online Incremental Learning***

$$a = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$b = \bar{y} - a\bar{x}$$

$$S_{x_n} = \sum_0^{n-1} x_i \quad S_{xx_n} = \sum_0^{n-1} x_i^2$$

$$S_{y_n} = \sum_0^{n-1} y_i \quad S_{xy_n} = \sum_0^{n-1} x_i y_i$$

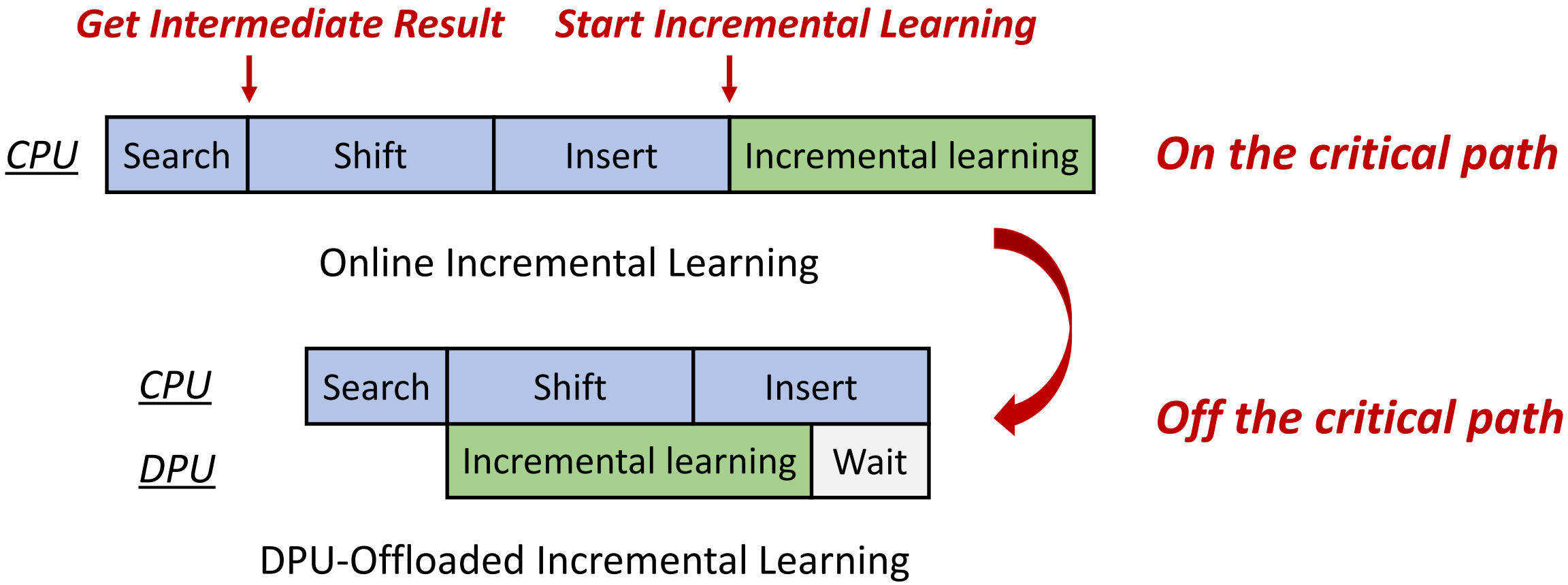
Intermediate Results

$$a_{new} = \frac{(n+1)S_{xy_{n+1}} - S_{x_{n+1}}S_{y_{n+1}}}{(n+1)S_{xx_{n+1}} - (S_{x_{n+1}})^2}$$

$$b_{new} = \frac{S_{xx_{n+1}}S_{y_{n+1}} - S_{x_{n+1}}S_{xy_{n+1}}}{(n+1)S_{xx_{n+1}} - (S_{x_{n+1}})^2}$$

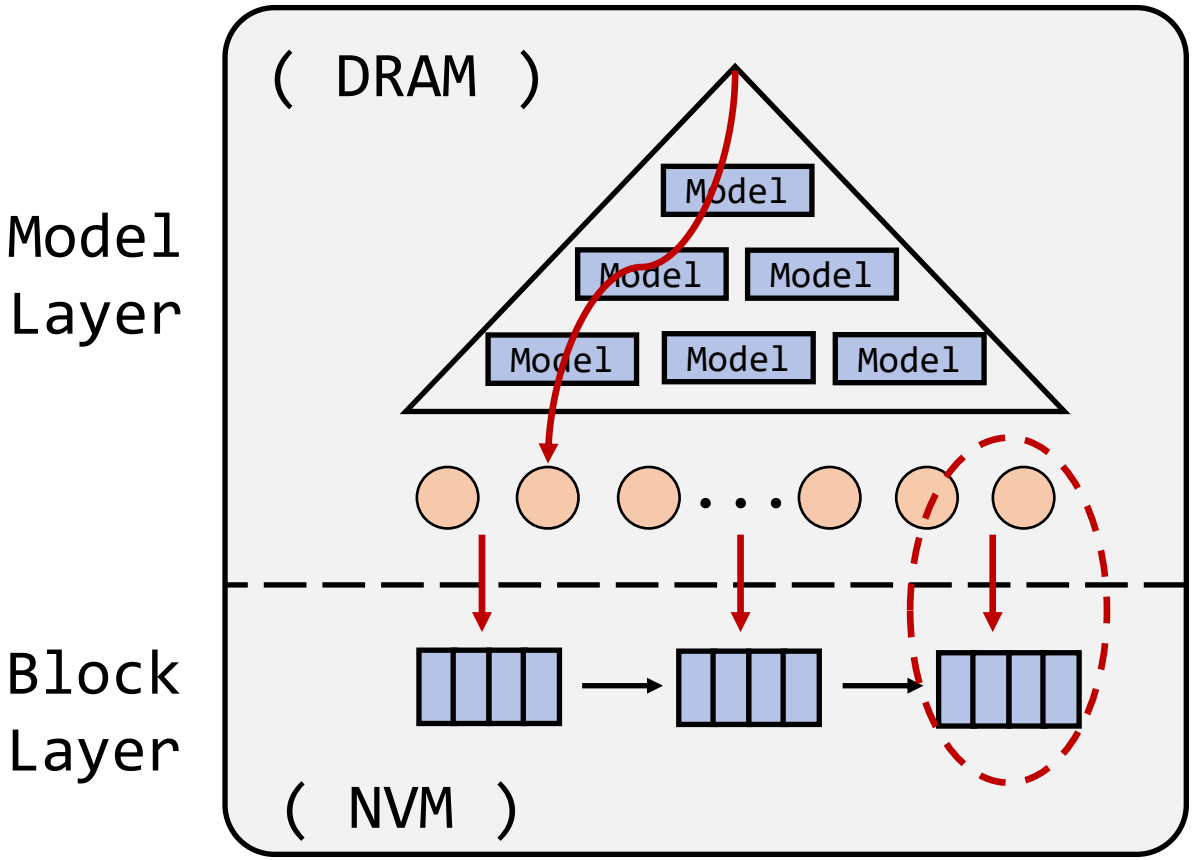
DPU-Offloaded Incremental Learning

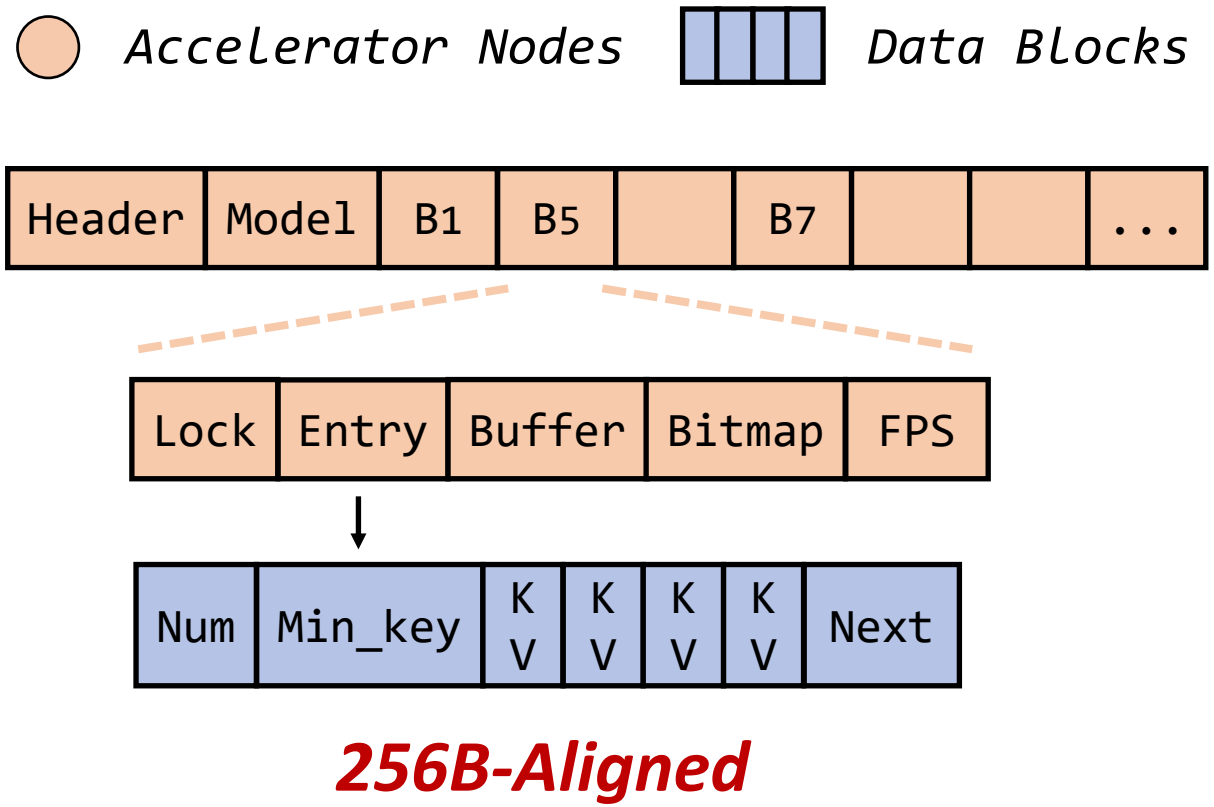
- Online incremental learning scheme is still on the ***critical path***.
  - Further ***offload*** incremental learning to the DPU side.



NVM-Friendly Index Structure

- ***DRAM-Accelerated Model Layer***: reduce NVM access.
- ***NVM-Aware Block Layer***: minimize NVM amplification.

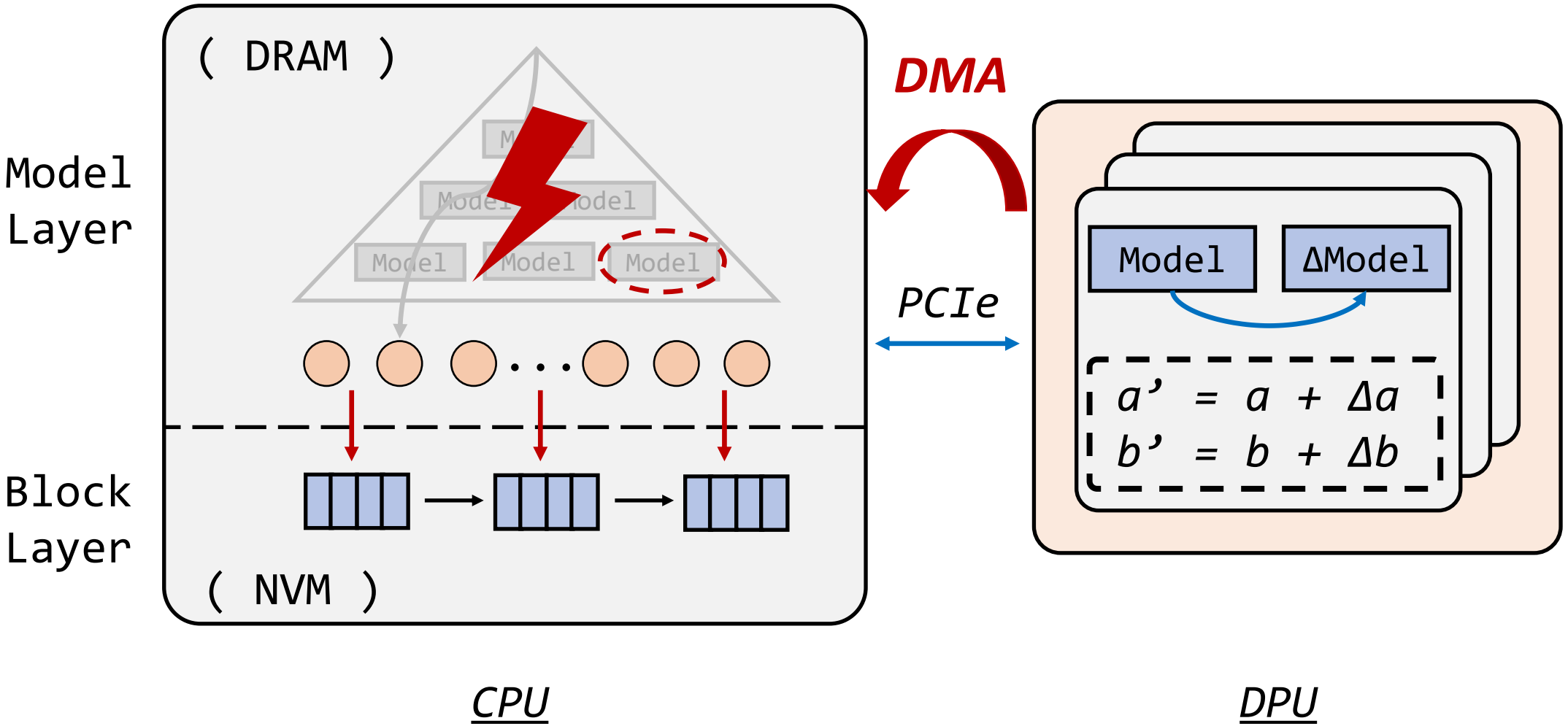




256B-Aligned

DPU-Assisted Instant Recovery

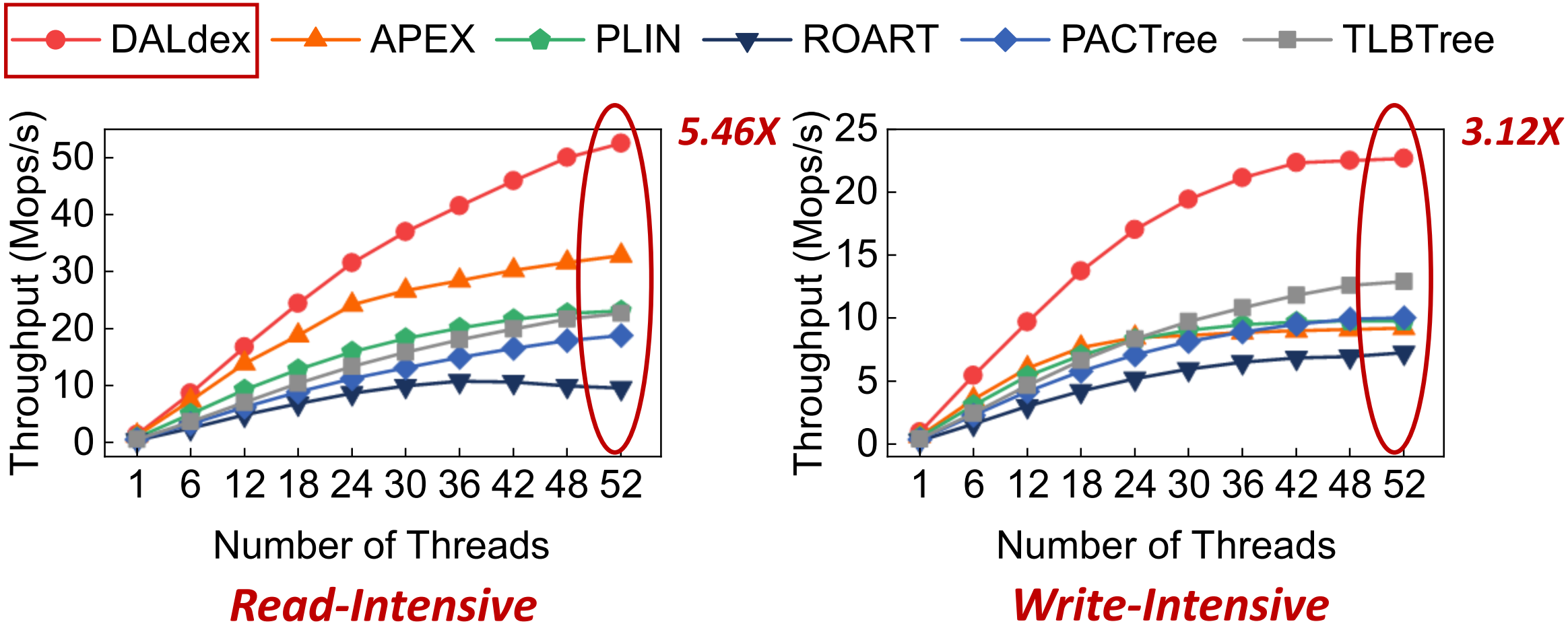
- The offloaded model structure naturally serves as a ***model backup***.



Experimental Setup

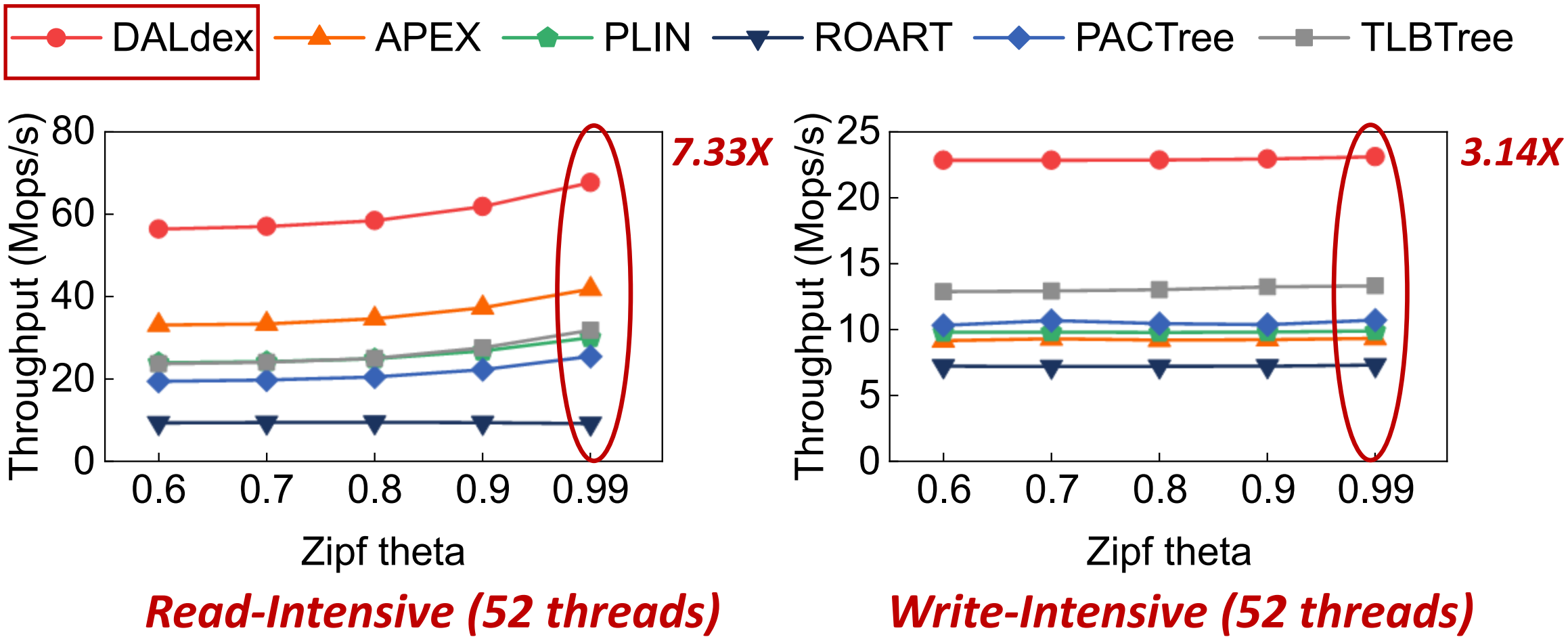
- **Testbed**
  - Intel(R) Xeon(R) Gold 6230 CPU @ 2.00GHz
  - 384GB DRAM & 768GB Intel Optane DC PMEM
  - NVIDIA Mellanox BlueField-2 DPU
- **Workloads**
  - Real-World Datasets: Books, Genome, OSM
  - 200M 8B Integer Keys & Values
- **Comparisons**
  - Learned Index: APEX [VLDB'21], PLIN [VLDB'22]
  - Non-Learned index: TLBTree [ICDE'21], ROART [FAST'21], PACTree [SOSP'21]

Overall Performance



DALdex improves the overall throughput by ***1.07-6.34X***.

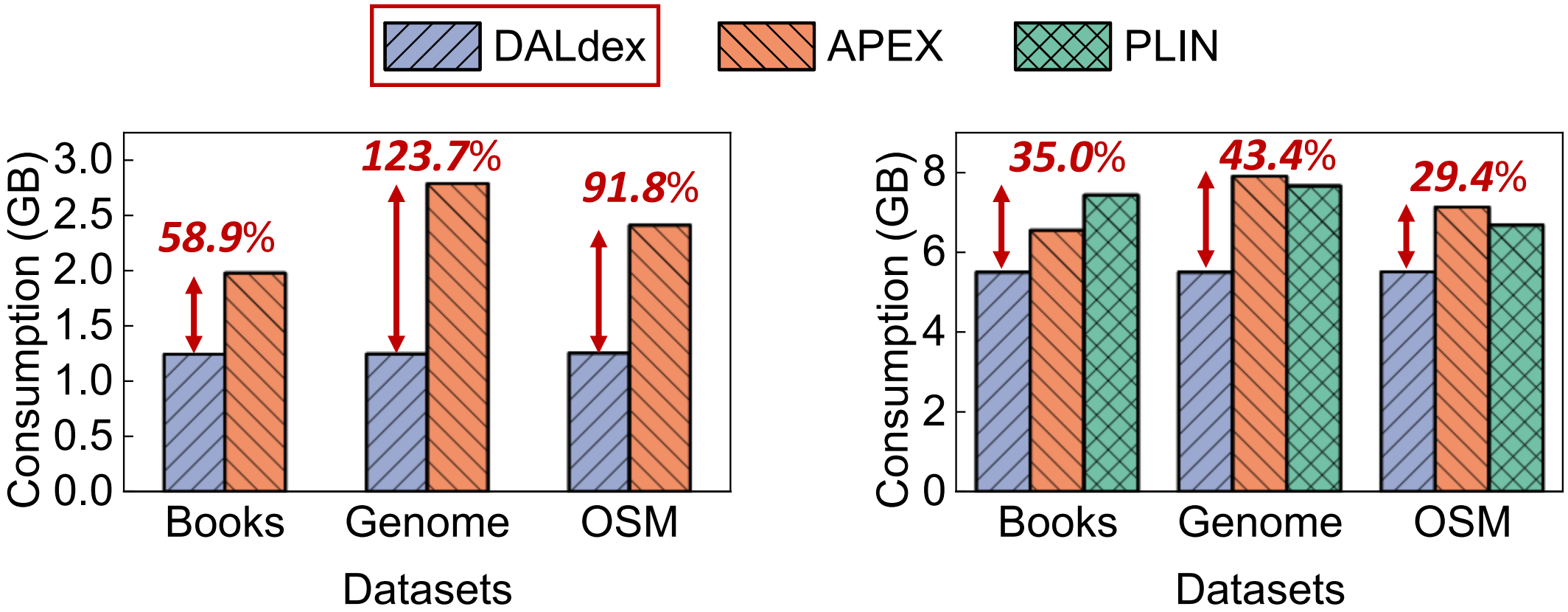
Skewed Workloads



Read-Intensive (52 threads) ***Write-Intensive (52 threads)***

DALdex improves the throughput by ***1.11-7.70X*** under skewed workloads.

Memory Overheads



DRAM overhead

NVM overhead

DALdex reduces memory overheads by ***58.9-123.7%*** and ***18.9-43.4%***.

Summary

- Existing persistent learned indexes suffer from ***expensive model retraining*** and ***excessive NVM access***.
- DALdex: A DPU-Accelerated Persistent Learned Index via Incremental Learning
  - ***DPU-Offloaded Incremental Learning***
  - ***NVM-Friendly Index Structure***
  - ***DPU-Assisted Instant Recovery***
- DALdex significantly outperforms state-of-the-art persistent indexes with minimal DRAM and NVM overheads.

Open Source Code: *https://github.com/CitySkylines/DALdex*

Thanks!

Q & A

Aoyang Tong, Yu Hua, Menglei Chen

Huazhong University of Science and Technology

Email: *aoyangtong@hust.edu.cn*