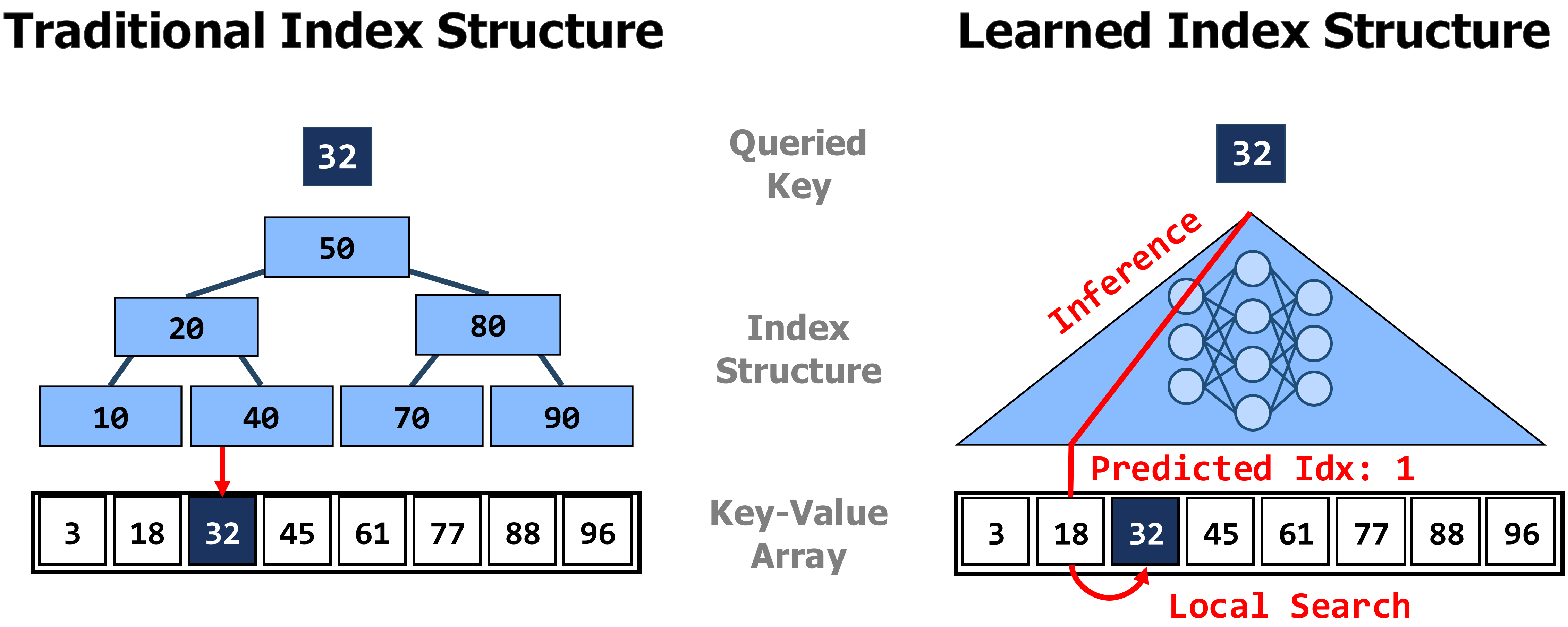
Accelerating String-key Learned Index Structures via Memoization-based Incremental Training

Minsu Kim
Jinwoo Hwang
Guseul Heo
Seiyeon Cho
Divya Mahajan†
Jongse Park

KAIST
Georgia Institute of Technology†

VLDB2024
GUANGZHOU

Learned Index Structure

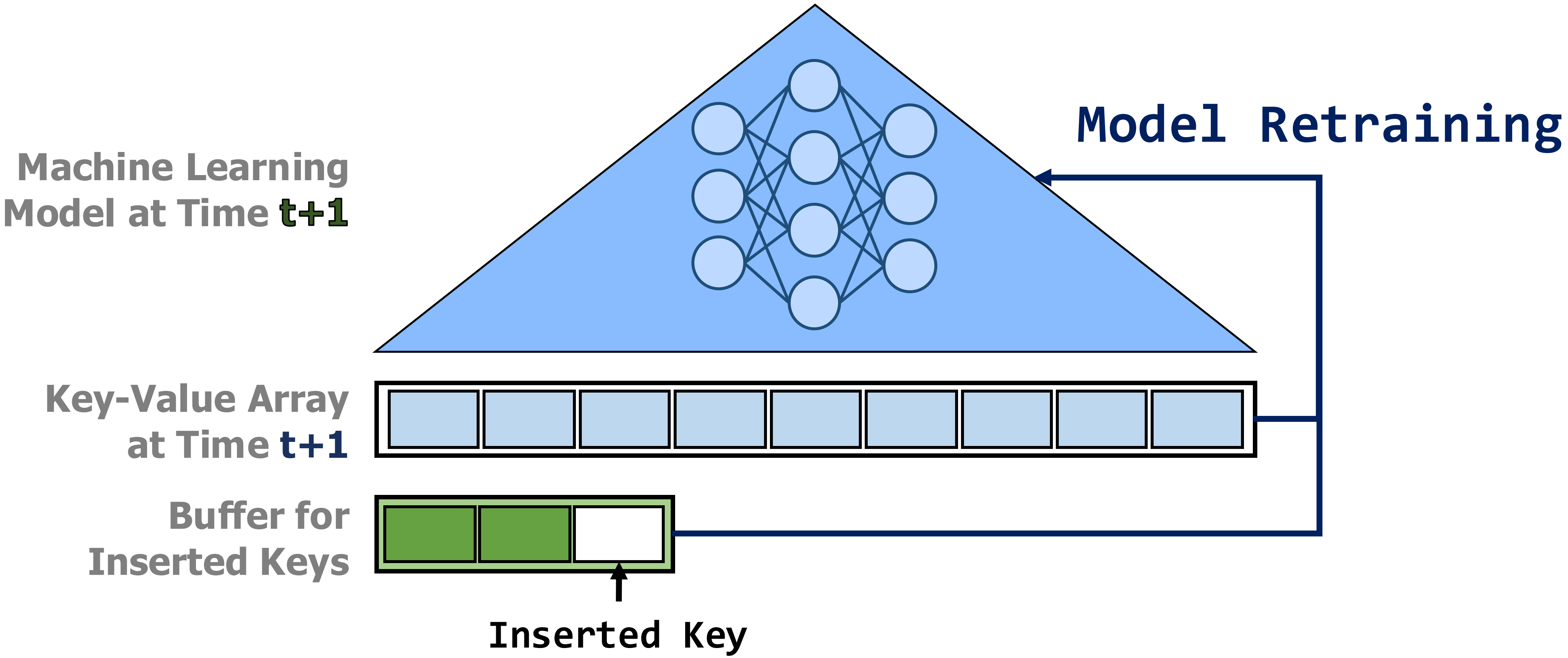


Learned Index Structure

	Traditional Index	Learned Index
Time Complexity	▲	▼
Performance	▼	▲
Index Size	▲	▼

- **Example Applications**
  - **Database**: BOURBON (2020)
    Learned Bigtable (2020)
  - **DNA Sequencing**: BLESS (2024)
  - **Embedded Sensor**: SENSORNETS (2023)

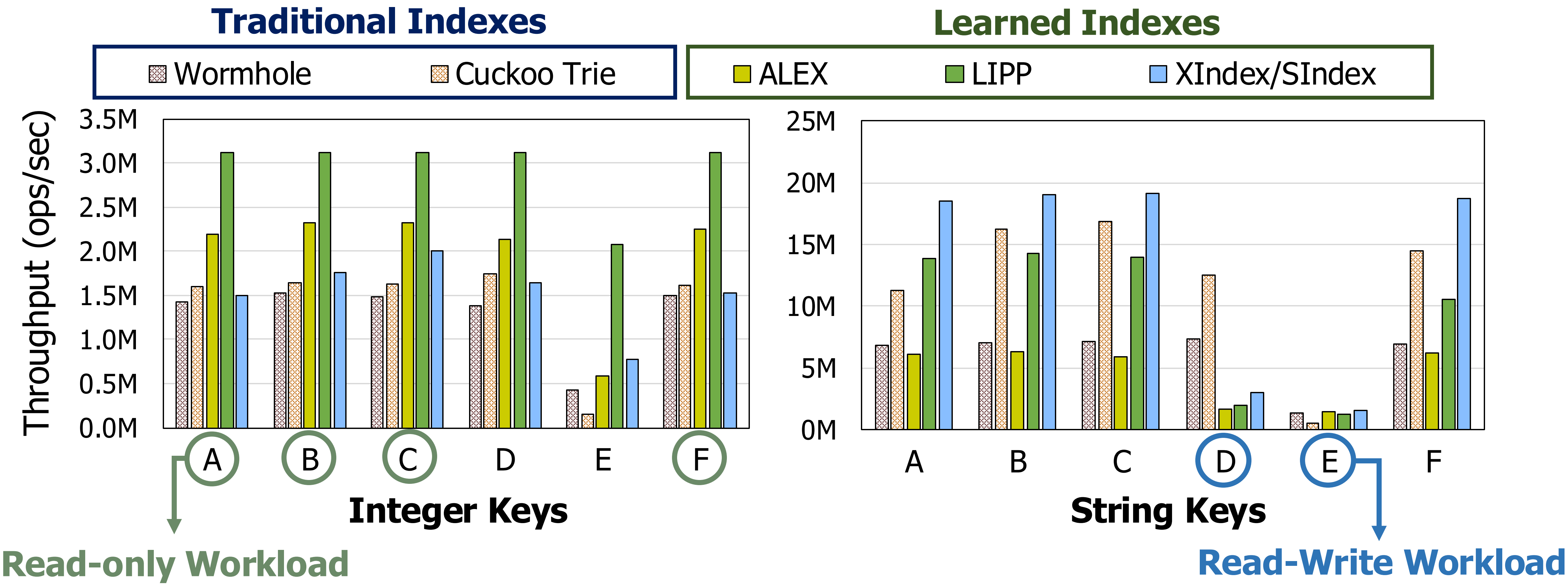
Updatable Learned Index



Updatable learned indexes require periodic retraining using the **entire keys**

Performance of Updatable Indexes

** Used YCSB (Yahoo Cloud Serving Benchmark) workloads*



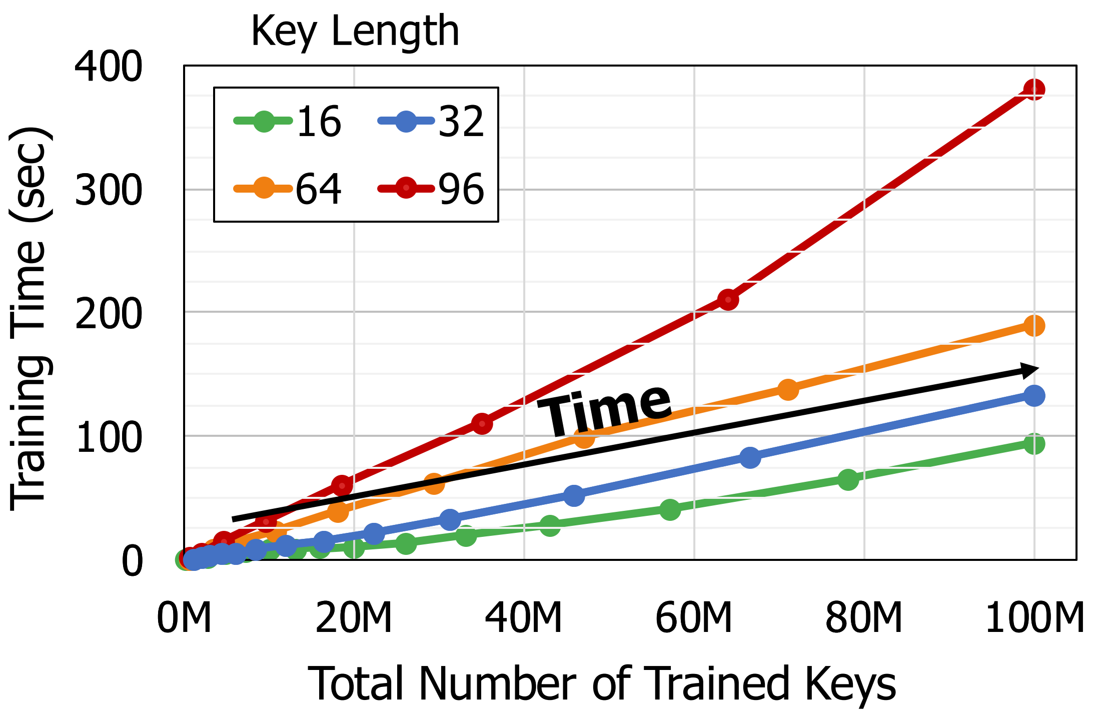
String-key learned indexes show **poor performance** for **read-write workloads**

Bottlenecks of Learned Index Training

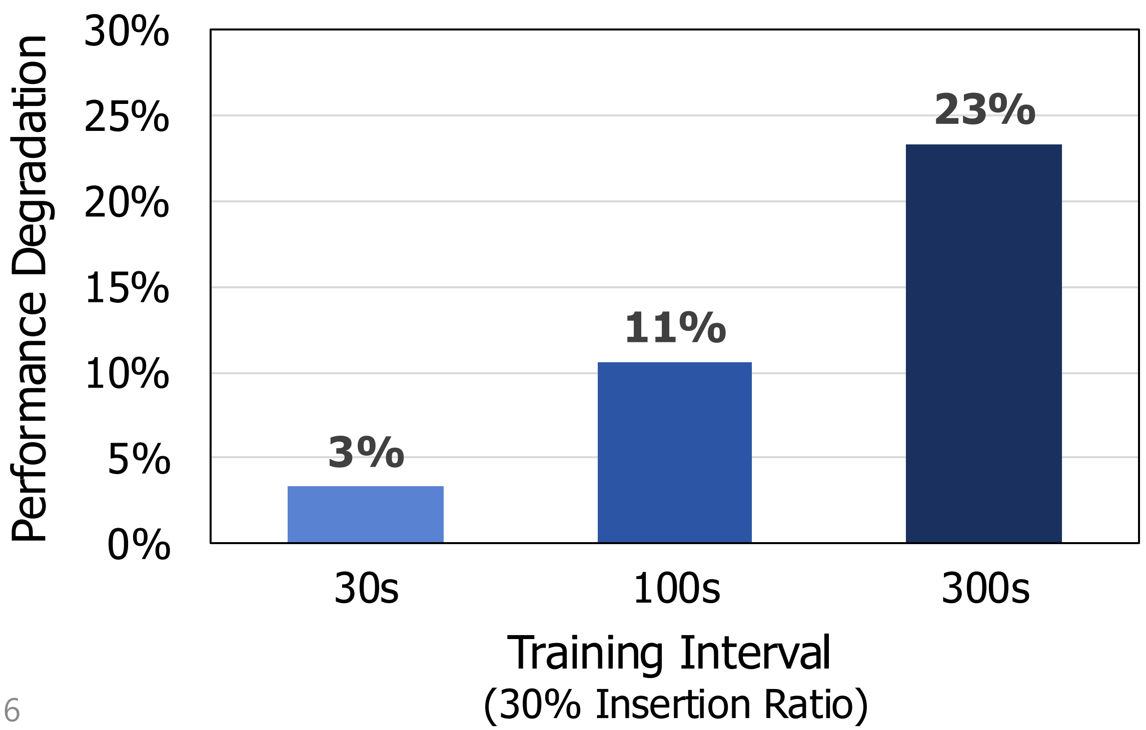
1. Bad scalability & performance due to accumulated keys

Accumulated keys **degrade the performance** of learned index
by delaying updates of ML model

Increasing Training Time



Performance Degradation with Slow Training



Bottlenecks of Learned Index Training

2. QR Decomposition Operations are Expensive

- Most learned indexes use **linear regression** for their ML model
- Solving linear regression involves **QR decomposition**

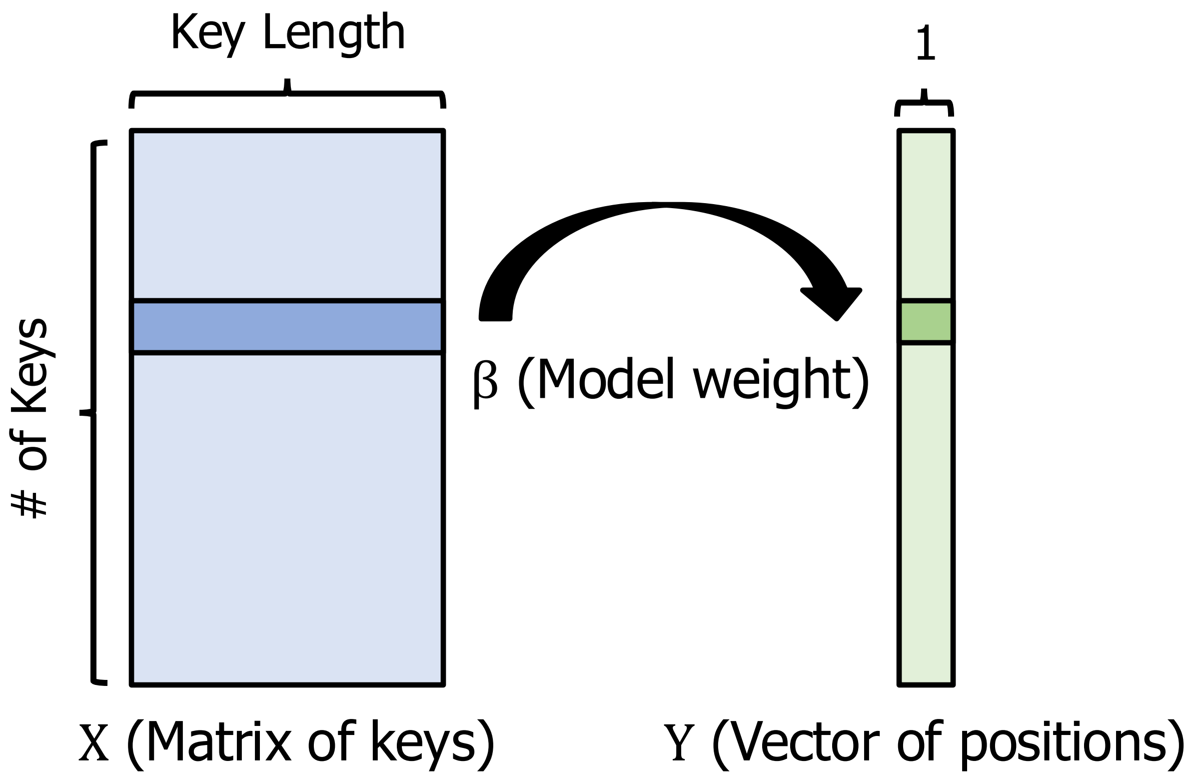
Linear Regression Model

$$X\beta = Y$$

Linear Regression Solution

$$\beta = \left(\mathbf{R}^{-1}\mathbf{R}^{-1^{T}}\right)X^{T}Y$$

, where $\mathbf{X} = \mathbf{QR}$



Bottlenecks of Learned Index Training

2. QR Decomposition Operations are Expensive

- **QR decomposition** is the major bottleneck when training
- **R Inverse and GEMM** are the second longest

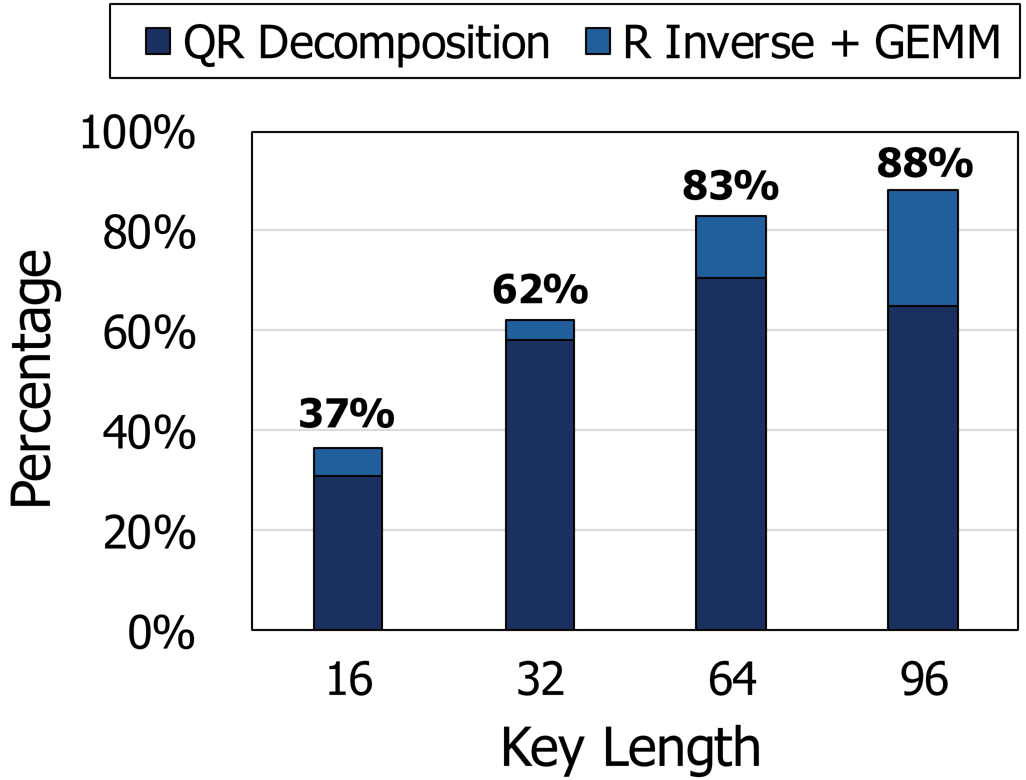
Linear Regression Model

$$X\beta = Y$$

Linear Regression Solution

$$\beta = \left(\mathbf{R}^{-1}\mathbf{R}^{-1^{T}}\right)X^{T}Y$$

, where $\mathbf{X} = \mathbf{QR}$



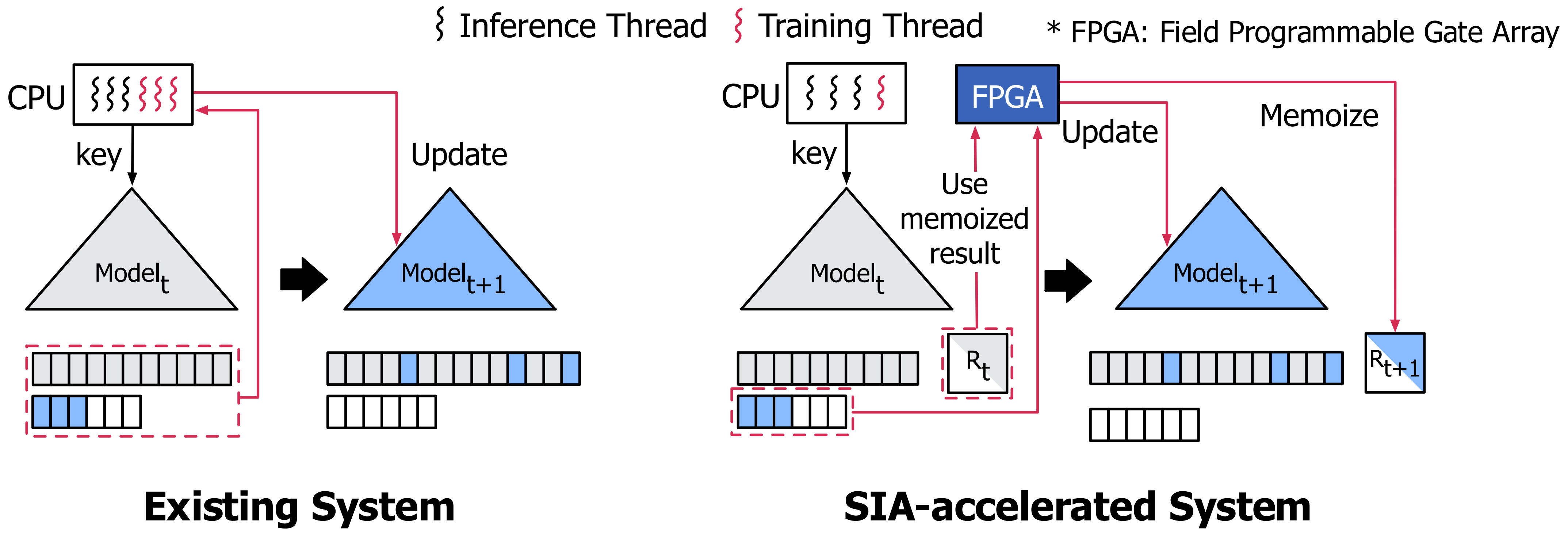
Existing String-key Learned Index Systems Offer Limited Performance

SIA: System Overview

Algorithm-Hardware Co-designed String-key Learned Index System

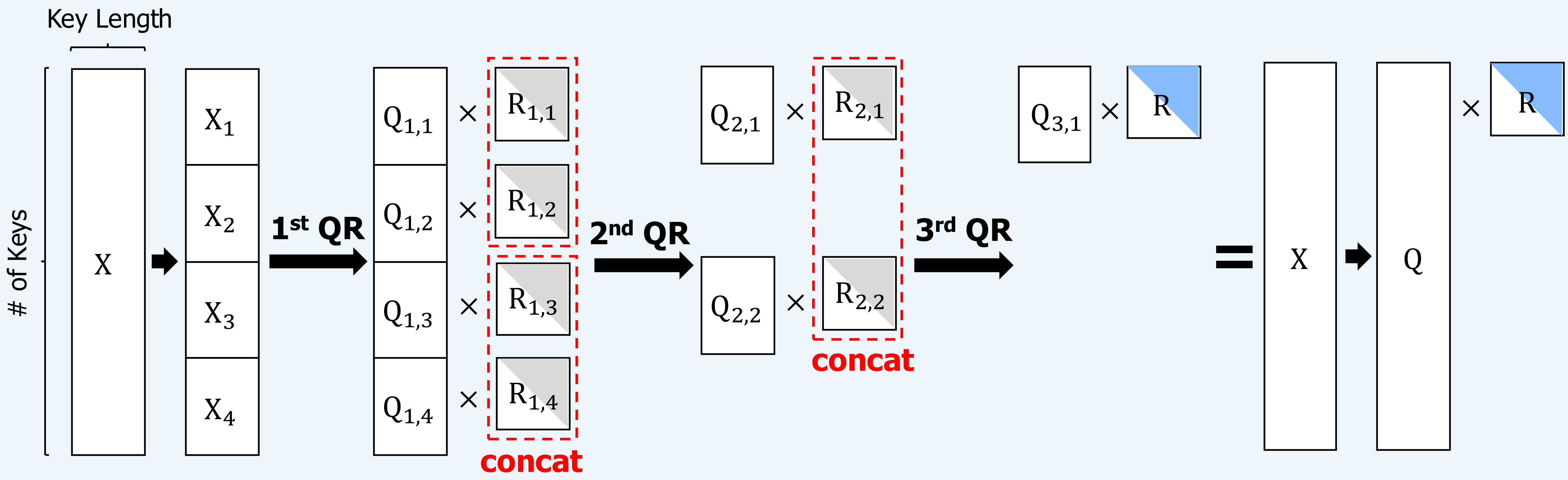
① **Algorithm** that reuses memoized intermediate results

② **Hardware** that offloads index training with FPGA accelerator



Insight from Parallel QR Decomposition

- Existing parallel QRD offers advantage to **tall-and-skinny** matrices
- Parallel QRD ensures **mathematical equivalence**



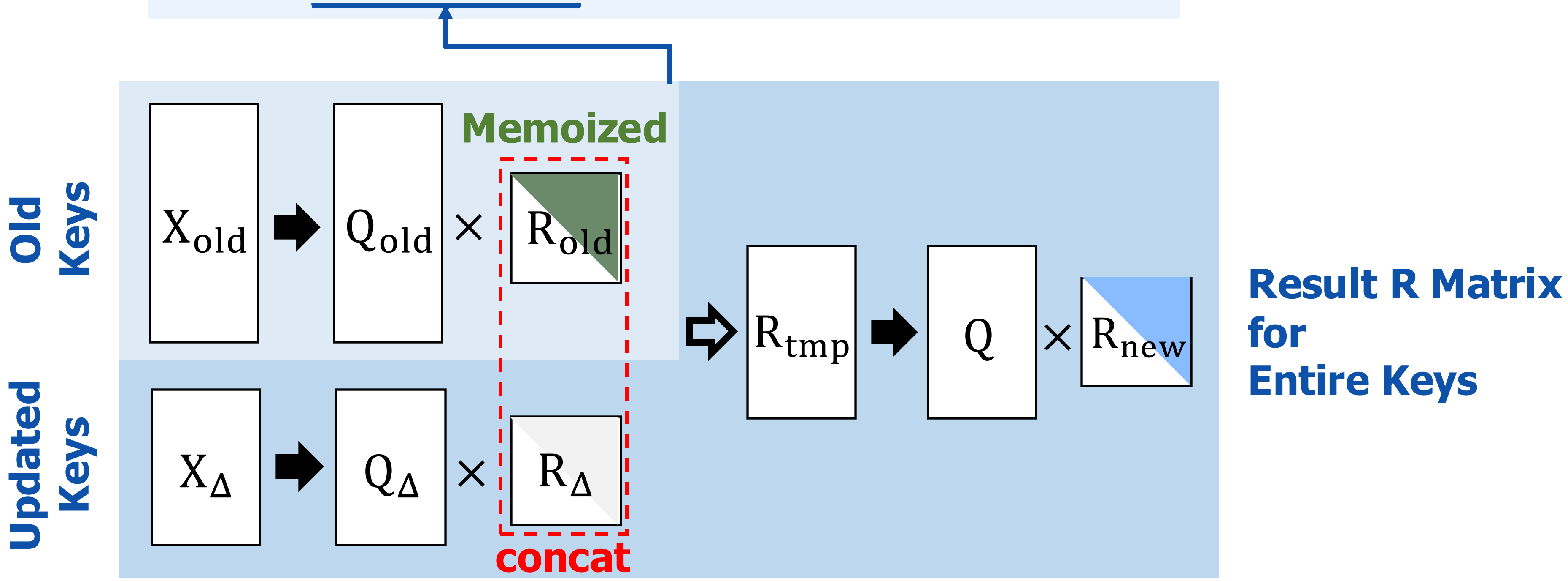
Algorithm Design

Incremental Index Learning

- Incremental index learning **reduces costly QRD** via memoization

Linear Regression Solution

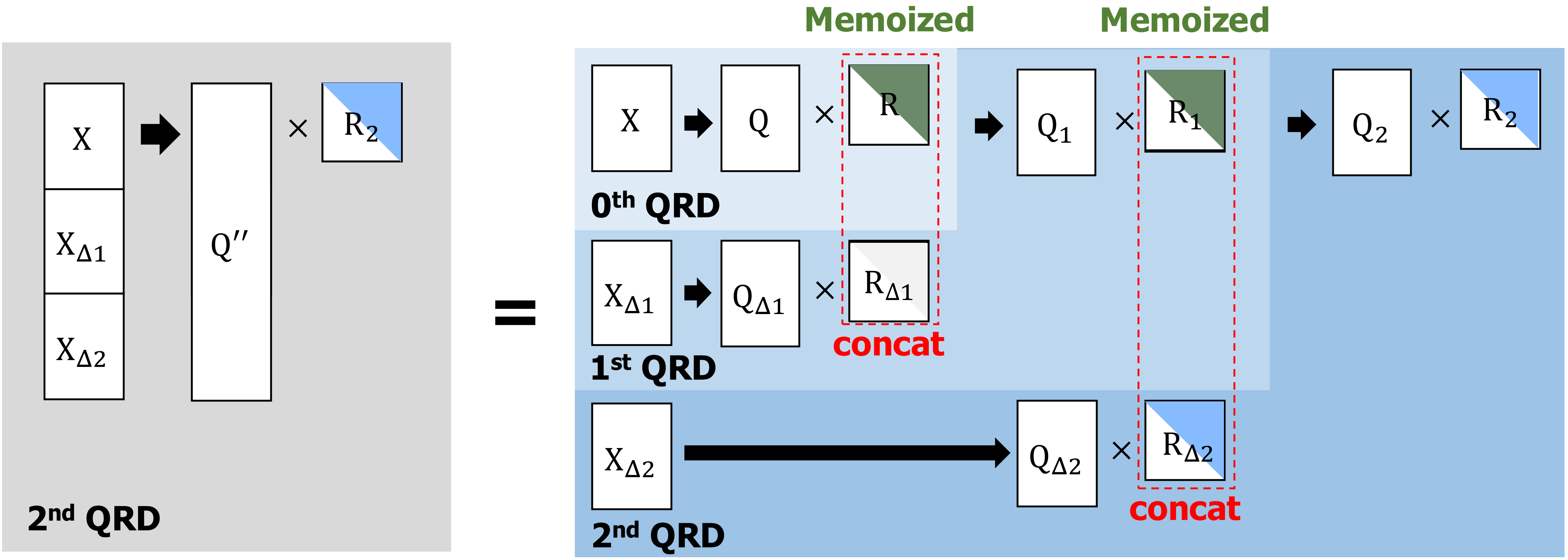
$$\beta = (\mathbf{R}^{-1}\mathbf{R}^{-1^{T}})X^{T}Y \text{ where } \mathbf{X} = \mathbf{QR}$$



Algorithm Design

Incremental Index Learning

- There is no need to perform QRD for entire key matrix

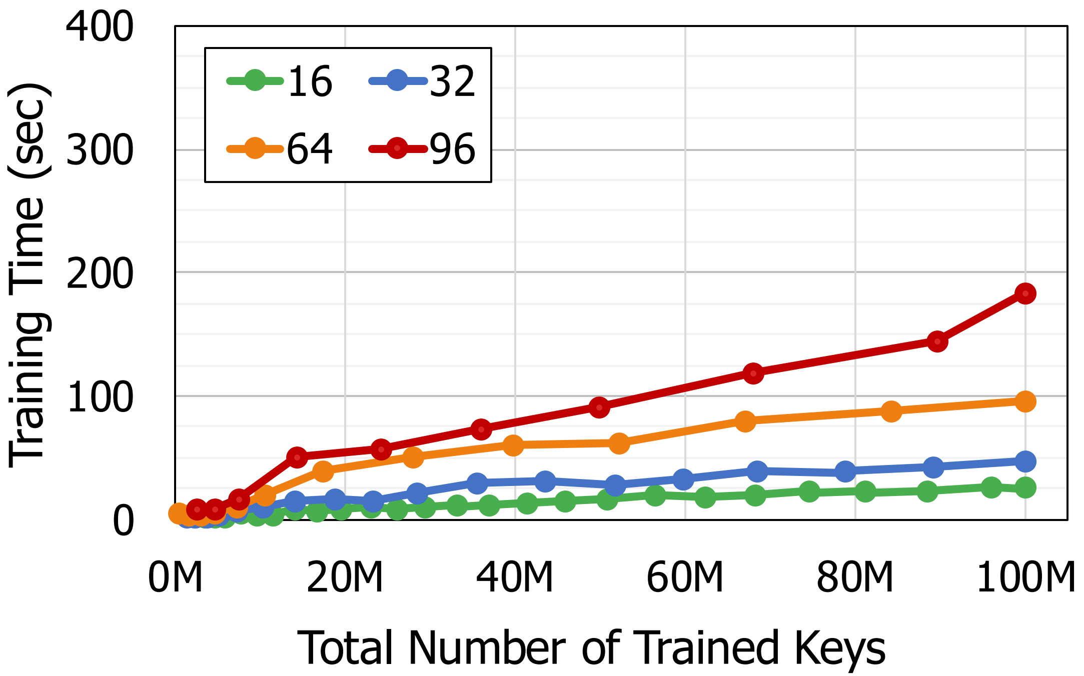


Naive QR Decomposition | **Memoized QR Decomposition**

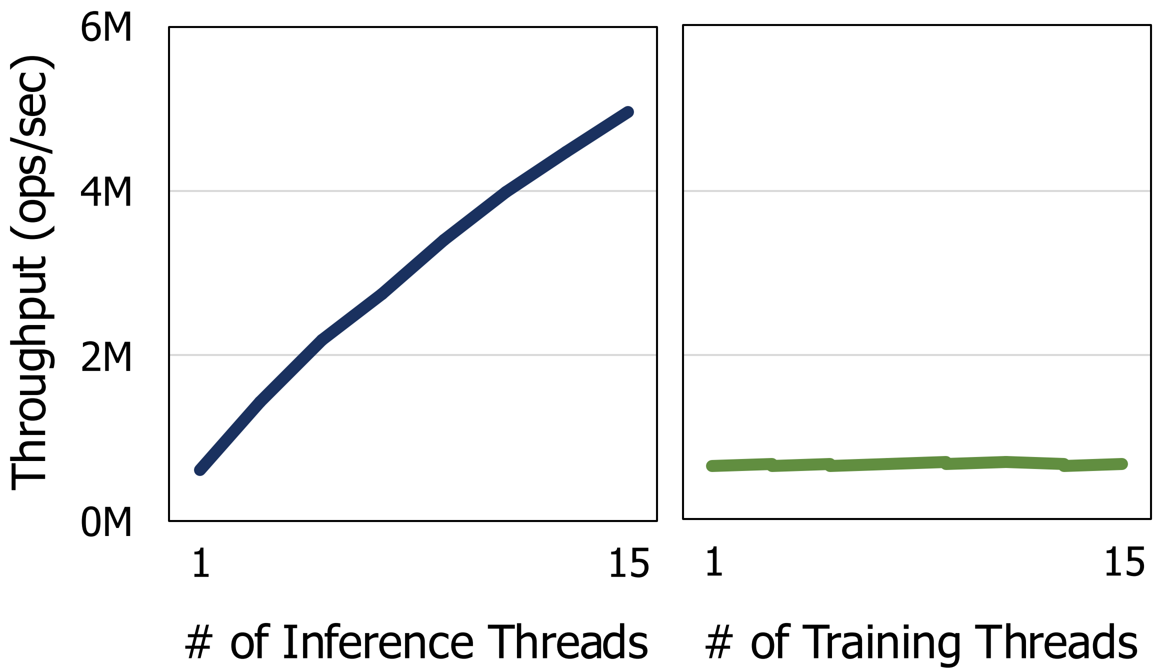
Why Do We Need Hardware Acceleration?

CPU-only solution is still slow due to **low efficiency in training**

Training Time with Incremental Learning

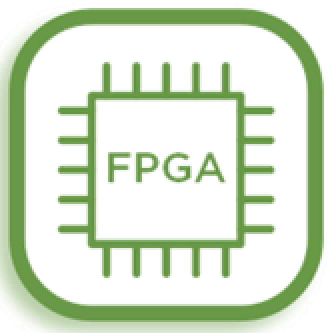


Throughput with Varying CPU Threads



Hardware Design

Hardware Selection: FPGA



Field Programmable Gate Array

- **Reconfigurable**
  Reprogrammable without changing hardware
- **Customizable**
  Programmable with user custom hardware logic
- **Parallelizable**
  Simultaneous operation of multiple logic blocks
- **Area & Energy Efficient**
  High performance at low operating cost

Hardware Design

FPGA Accelerator Architecture

Linear Regression Solution

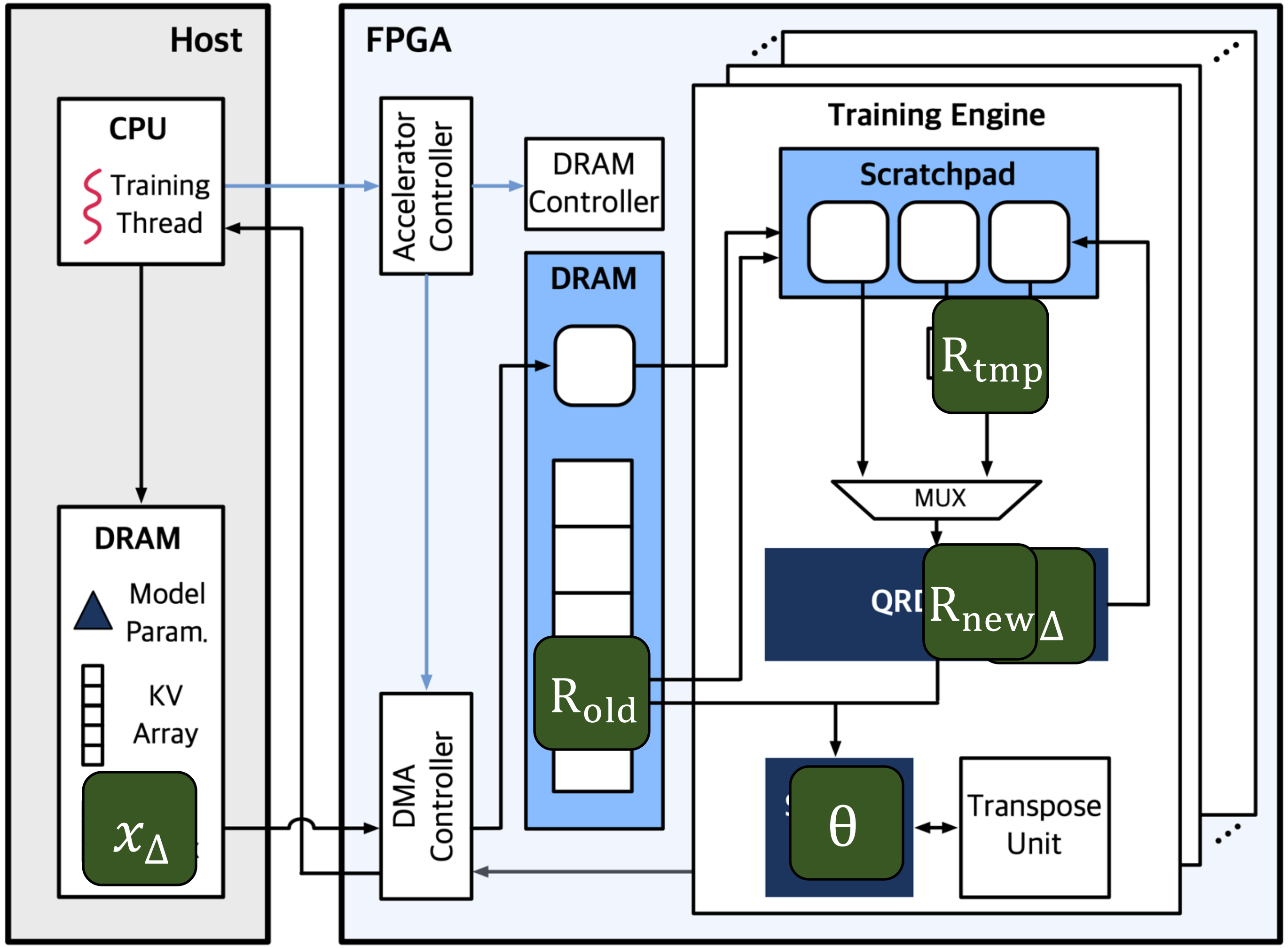
$\beta = (\mathbf{R}^{-1}\mathbf{R}^{-1^{T}})X^{T}Y$ where $\mathbf{X} = \mathbf{QR}$

FPGA accelerator calculates

$$\boldsymbol{\theta} = \left(\mathbf{R}^{-1}\mathbf{R}^{-1^{T}}\right)$$

with incremental index learning

Calculation result is returned to host CPU



Evaluation Methodology

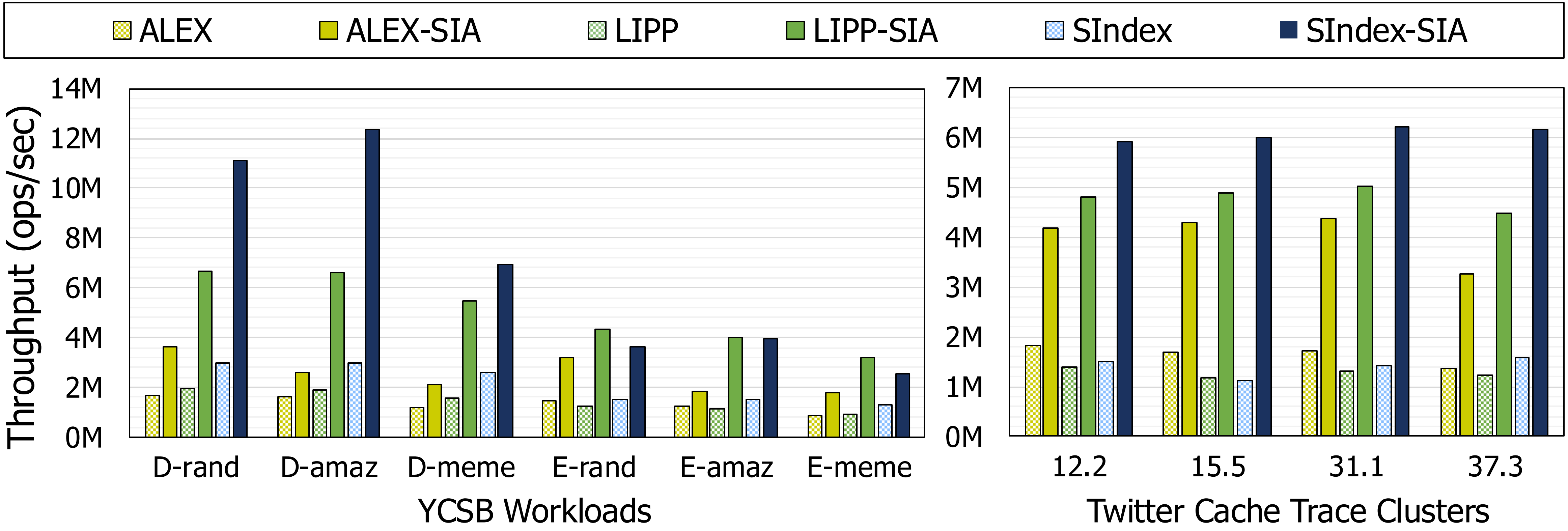
- **Baselines**
  - Wormhole[1]
  - Cuckoo Trie [1]
  - SIndex [2]
  - ALEX [2]
  - LIPP [2]

  [1] Traditional indexes
  [2] Updatable learned indexes

- **FPGA**
  - Intel Arria 10 GX-1150 (Synthesized to 272MHz)

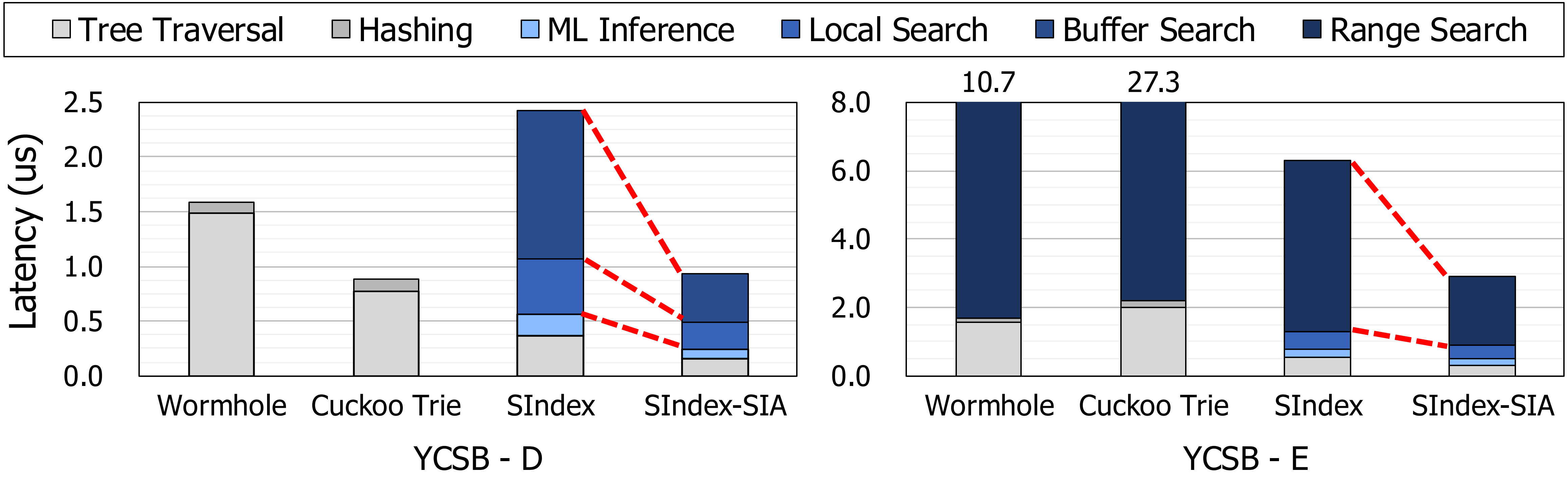
Dataset	Workload	
"amaz" Amazon review dataset	**YCSB – D** Read & Insert queries	**YCSB – E** Range & Insert queries
"meme" Memetracker dataset		
"rand" Randomly generated strings		
Twitter Cache Trace **12.2, 15.5, 31.1, 37.3**	**Twitter Cache Trace** **12.2, 15.5, 31.1, 37.3** Read & Insert Queries	

Performance Evaluation



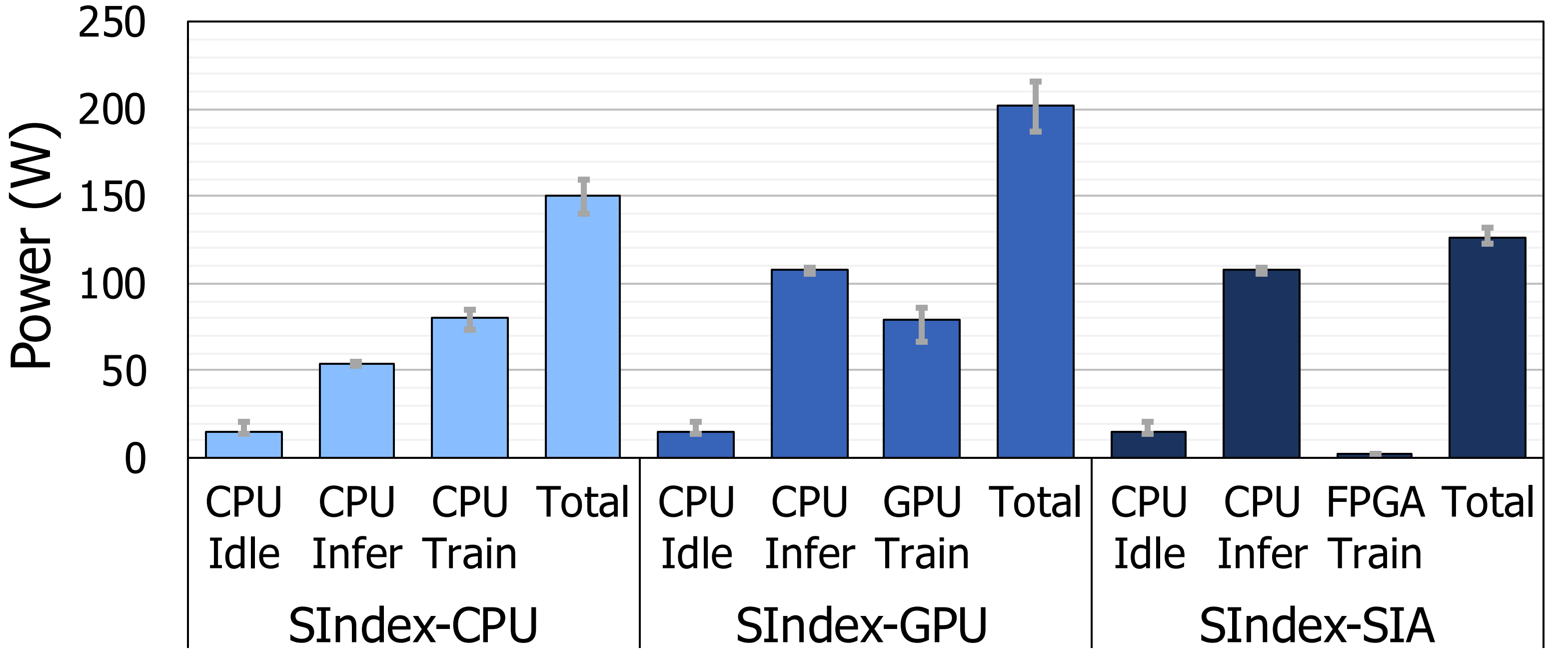
Learned indexes with SIA shows an average of **2.9x throughput improvement** compared to learned indexes without SIA

Latency Breakdown



Learned Index with SIA benefits from **reduced search time** due to "freshness" of learning model

Energy Efficiency Evaluation



	Normalized Performance per Watt
SIndex-CPU	1.00x
SIndex-GPU	1.67x
SIndex-SIA	**2.89x**

* CPU: Intel Xeon Gold 6226R
* GPU: NVIDIA RTX 2080 TI

SIA achieves higher energy efficiency with **low energy usage of FPGA**

(**28x** less than NVIDIA RTX 2080 TI GPU)

Suitable for continuous retraining of **learned index system**

More Results in Paper

- **Hardware Resource Utilization**
- **Memory Consumption Comparison**
- **Ablation Study**
- **Throughput with Different Query Distribution**
- **Implication of Lazy Delete Query Handling**

Conclusion

- **SIA**
  - Algorithm-hardware co-designed string-key learned index system
- **Contributions**
  - Identifies and mitigates bottleneck of current learned index structures
  - Accelerates model retraining via memoization-based algorithmic approach
  - FPGA-based hardware design further reducing the training time
- **Results**
  - **2.9x** higher throughput than learned indexes without SIA