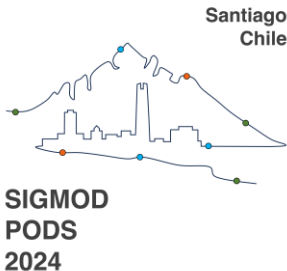


Can Learned Indexes be Built Efficiently?

A Deep Dive into Sampling Trade-offs

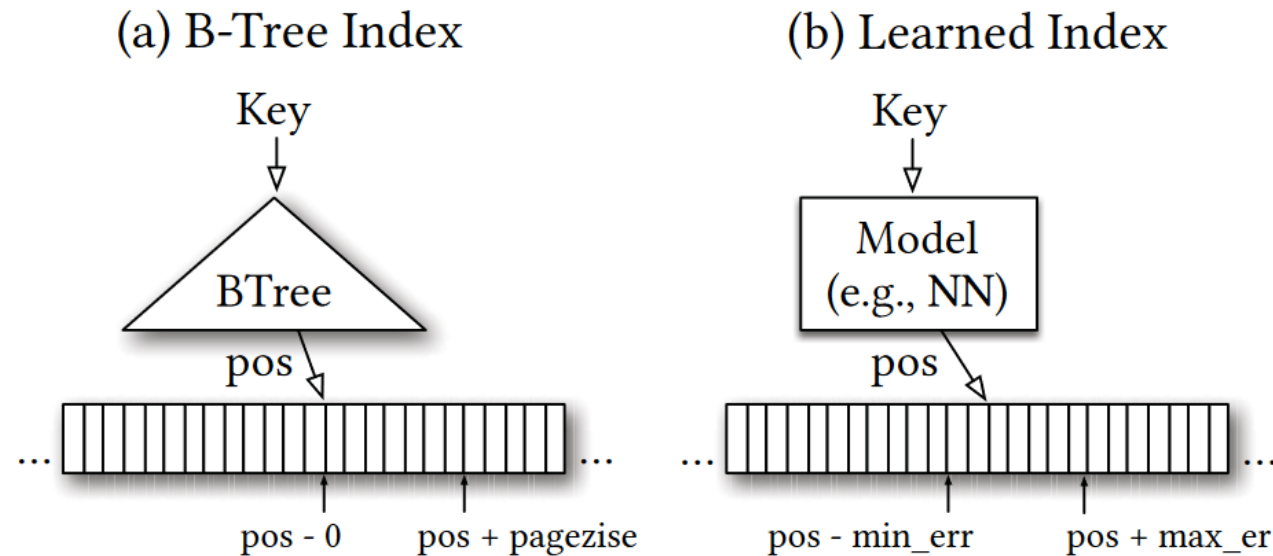
Minguk Choi, Seehwan Yoo, Jongmoo Choi

Dankook University, South Korea



1. Introduction

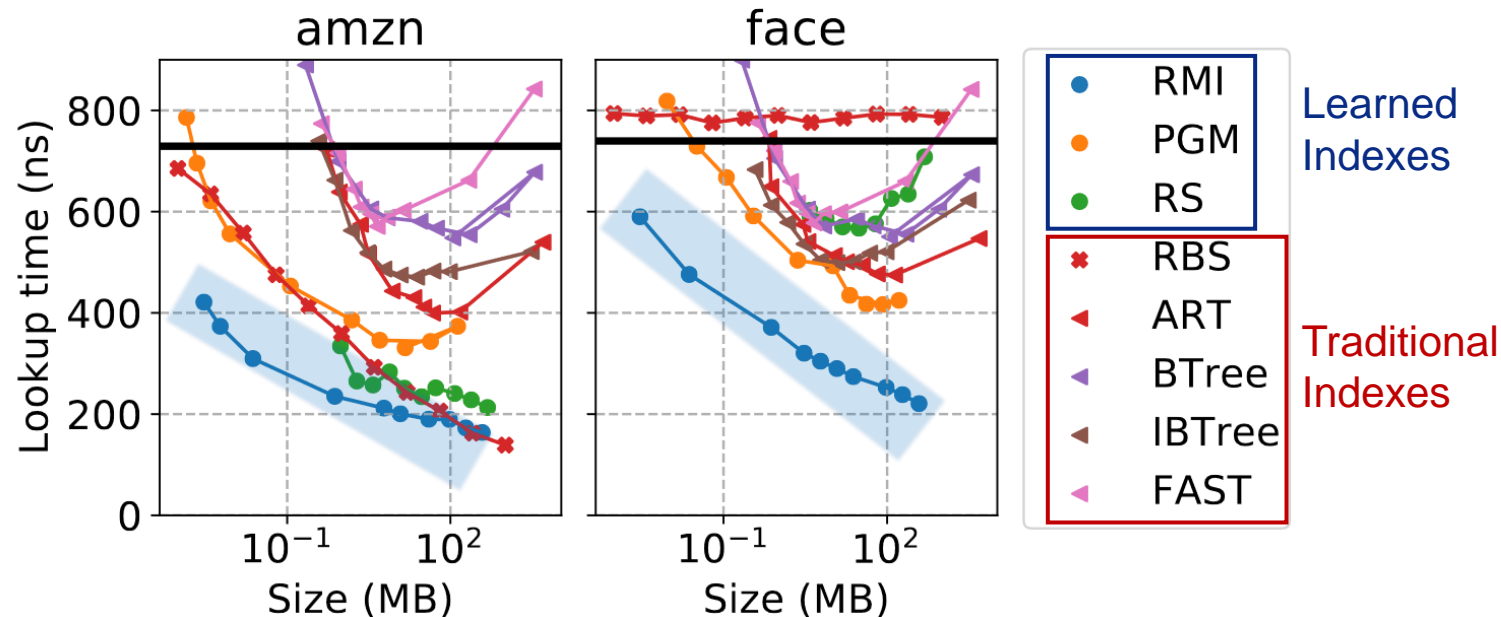
- Learned Index
 - Index structure employs machine learning techniques
 - View the index as a model that **predicts the position of a key** in a sorted array



(The Case for Learned Index Structures, SIGMOD '18)

1. Introduction

- Learned Index
 - **Space-efficient** by effectively compressing data distribution through the model
 - Pareto optimal in terms of index size and lookup latency in read-only workloads
 - No alternative exists that has both a smaller size and lower latency

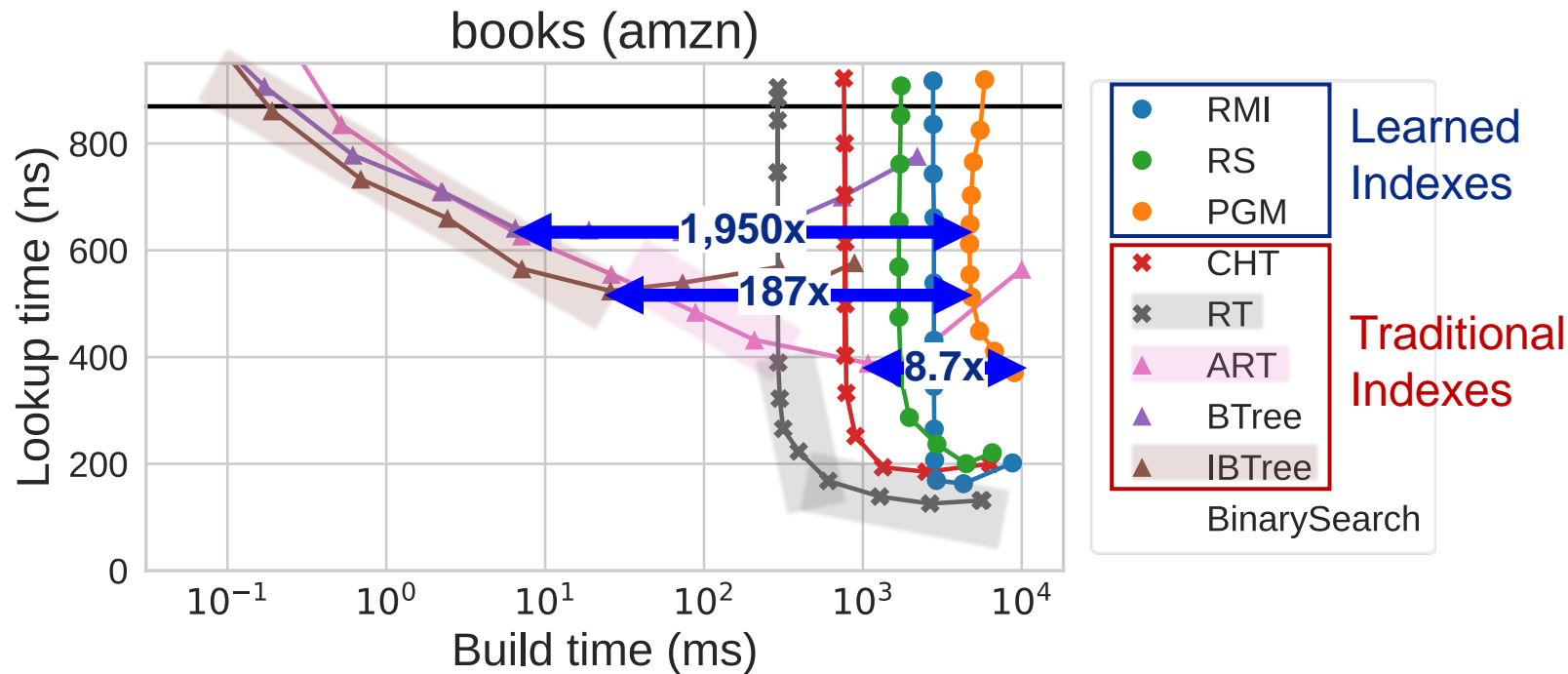


(Benchmarking Learned Indexes, VLDB '20)

1. Introduction

■ Limitation of Learned Index: **Long Index Build Time**

- Significantly (up to about **2,000x**) slower than traditional indexes
 - Not Pareto optimal (build-efficient) for build time and lookup latency
- Still, there are application (e.g., **LSM-Tree**) where the index build time is crucial



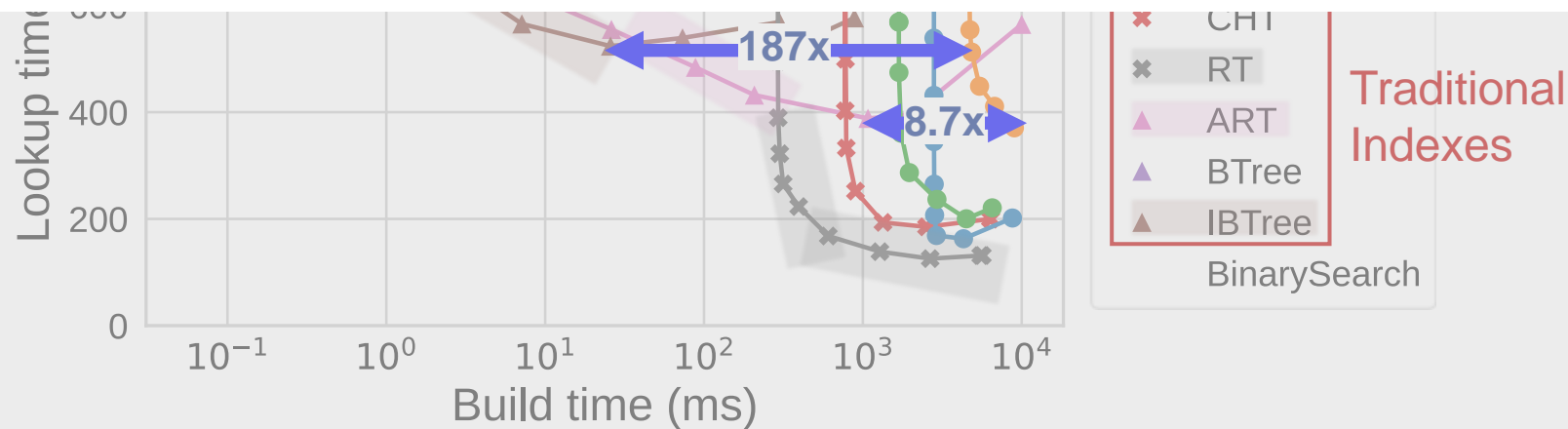
1. Introduction

- Limitation of Learned Index: **Long Index Build Time**
 - Significantly (up to about **2,000x**) slower than traditional indexes



Long build time has been identified as
a high priority for future work in various papers:

RMI (SIGMOD '18), RadixSpline (aiDM '20), PGM-Index (VLDB '20), SOSD (VLDB '20), Critical-RMI (VLDB '22)



1. Introduction

- Primary Reason for Long Build Time

$$\text{Index build time} = \text{Per - element overhead} \times \text{Number of elements}$$

1) **Higher** per-element training overhead

2) **Complete** traversal and training

- To Mitigate Per-element Overhead
 - **Light-weight** model: RadixSpline (aiDM `20), Bourbon (OSDI `20)
 - It **still** shows longer build time than traditional indexes

This study began with the question 🤔...

Since the learned index uses the model,

**Can't it learn efficiently even
with less data?**

1. Introduction

- Our Approach: **Sampling**

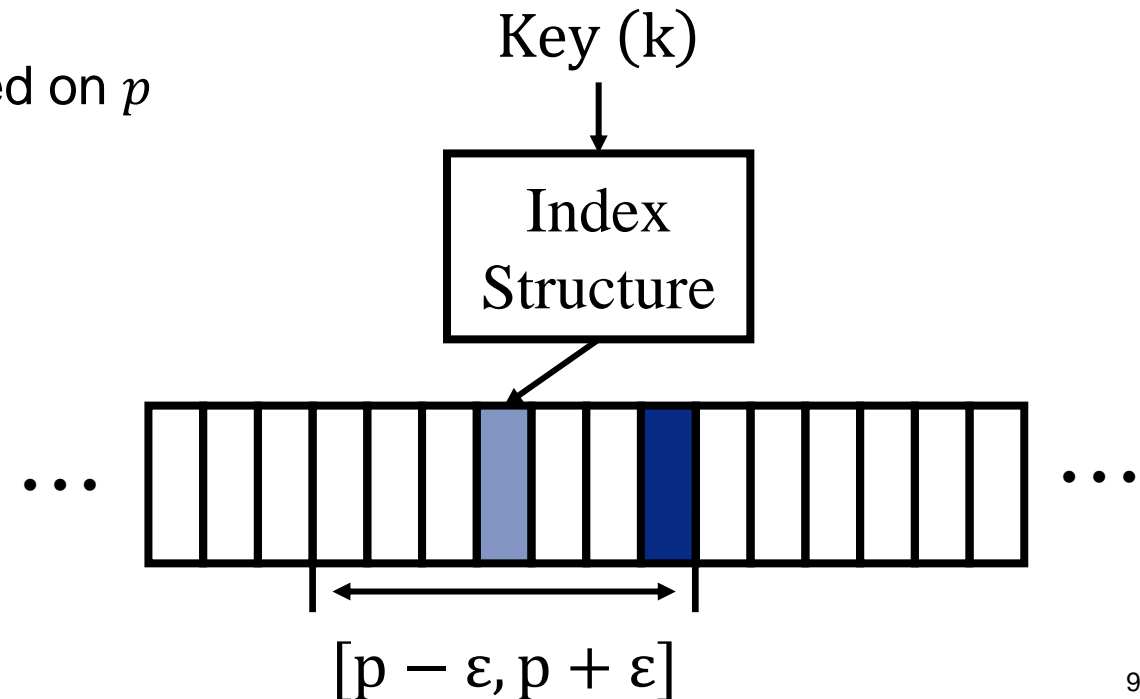
- While sampling may seem simple and even naïve, it is indeed complex

- Challenges

1. **Absence** of benchmark for sampling applied indexes
2. **Losing** the error-bound property due to sampling loss
3. **Complex** trade-offs in terms of model, index, and micro-architecture

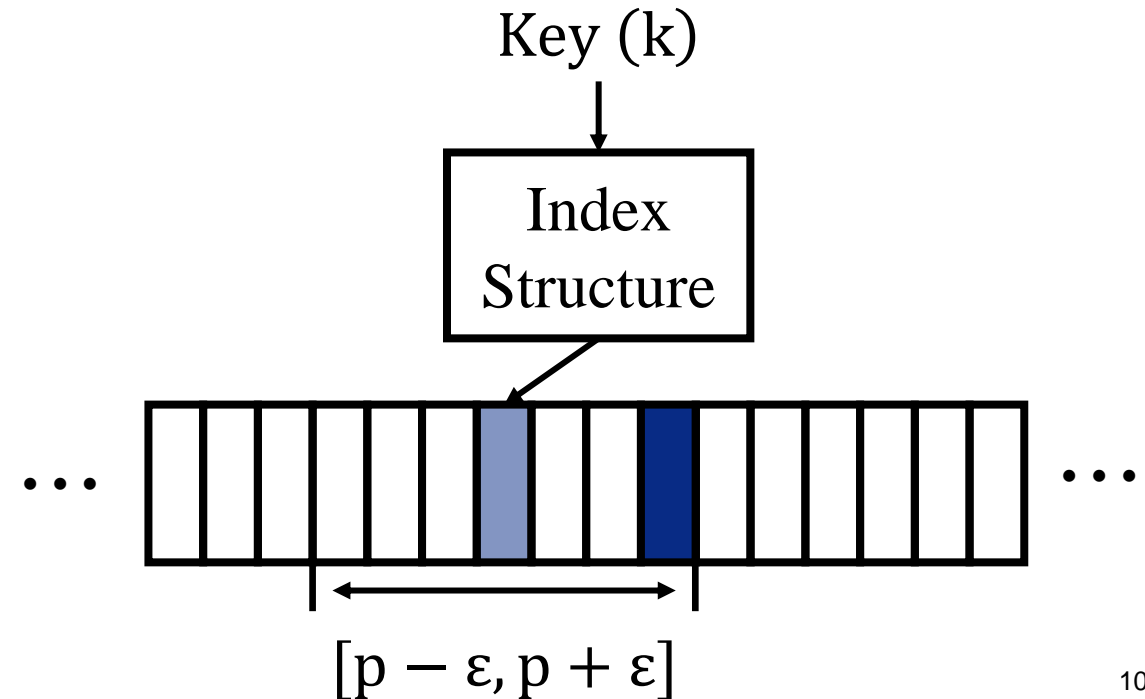
2. Background

- Workload: Read-only In-memory
 - Practical beginning point of learned index
 - Dataset (D): Sorted array of unique integer keys without duplicates
 - Lookup: Find the position of a lookup key k in D
 - ① Prediction: Estimate the position of k as p
 - ② Correction: Find the exact position of k based on p



2. Background

- Workload: Read-only In-memory
 - Error-bound property
 - $\forall k \in D, Error(k) = |Pred(k) - Pos(k)| \leq \varepsilon$ (*= error - bound*)
 - k exists in correction range ($= [p - \varepsilon, p + \varepsilon]$) \rightarrow binary search
 - Important for robustness,
 - Especially where correction is expensive
 - E.g., Disk or remote I/O environments



3. Design

1. Unified Sampling Algorithm & Implementation



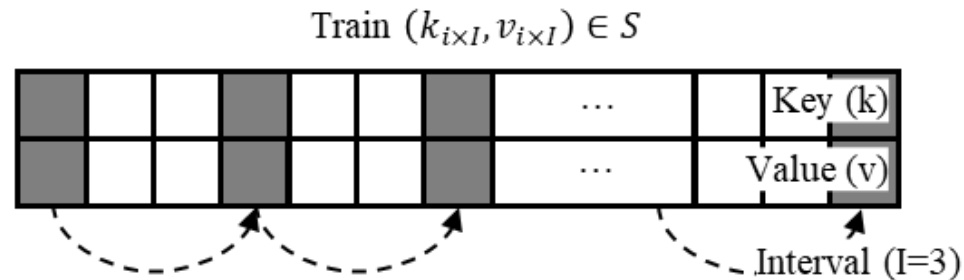
BASIL (Benchmark of Sampling Applied Learned Indexes)

1) Unified sampling algorithm

- **Systematic** sampling: extract every I^{th} (I = sampling interval) key from the first key to the last key
 - Pros: Simple, universal, no decision/reordering cost
 - Cons: Not optimal (other methods, e.g., adaptive, should be explored)

2) Unified sampling implementation

- All indexes access and train only sample key-value data from the entire dataset



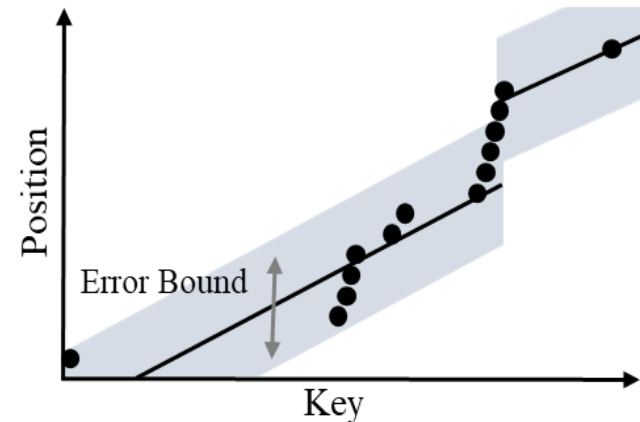
BASIL. Access I -th sample key-value in original dataset

3. Design

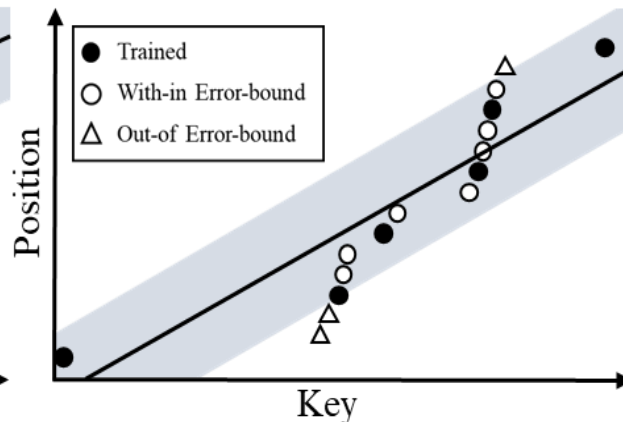
2. Sample Learning Algorithm

- EB-PLA (Error-bounded Piece-wise Linear Approximation)
 - Train **all** keys with error-bound $\varepsilon \rightarrow \text{Error}(k) \leq \varepsilon$
 - Train the **sample I^{th}** keys with error-bound $\varepsilon \nrightarrow \text{Error}(k) \leq \varepsilon$
 - **Loss** of the error-bound property, which is learning objective of the model

(a) Train All ($I = 1$)
with $\varepsilon = 3$



(b) Train Sample ($I = 3$)
with $\varepsilon = 3$



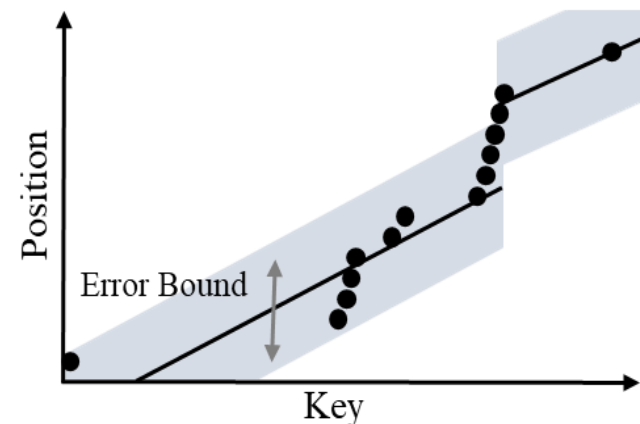
3. Design

2. Sample Learning Algorithm

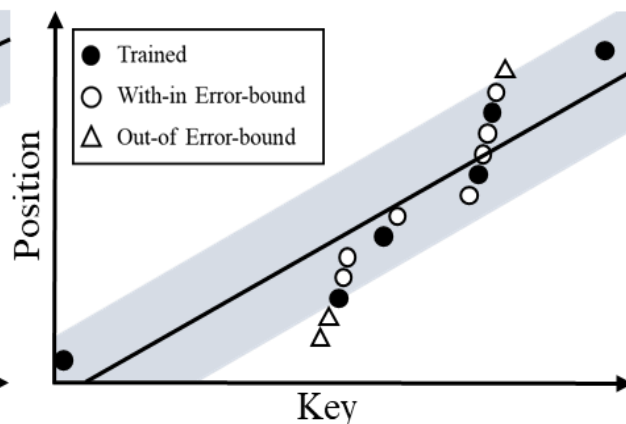
- Sample EB-PLA

- Refine the error-bound due to sampling loss to $\varepsilon' (= \varepsilon + I - 1) \rightarrow Error(k) \leq \varepsilon'$
 - Preserve the error-bound property, but cannot guarantee the desired error-bound (ε)
- Replace the learning error-bound to $\delta (= \varepsilon - I + 1) \rightarrow Error(k) \leq \varepsilon$
 - Preserve the error-bound (ε) by learning less data with a smaller and stricter error bound

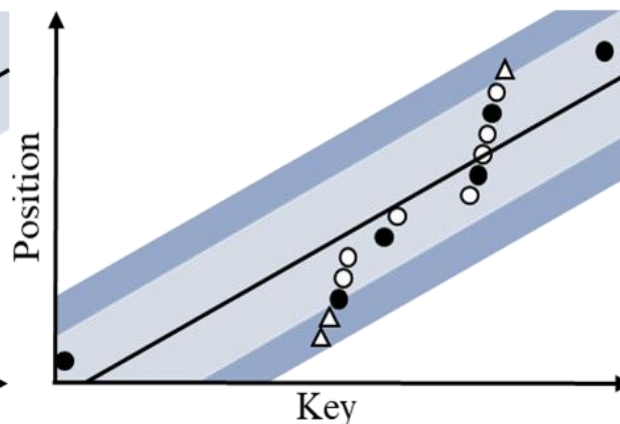
(a) Train All ($I = 1$)
with $\varepsilon = 3$



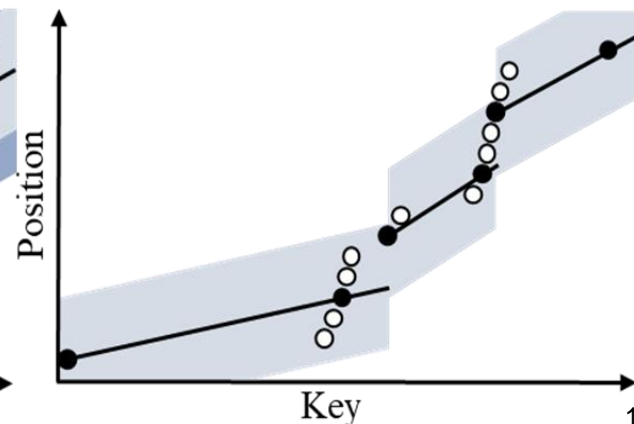
(b) Train Sample ($I = 3$)
with $\varepsilon = 3$



(c) Refine the Error-bound
 ε from 3 to 5



(d) Train Sample ($I = 3$)
with $\delta = 1$

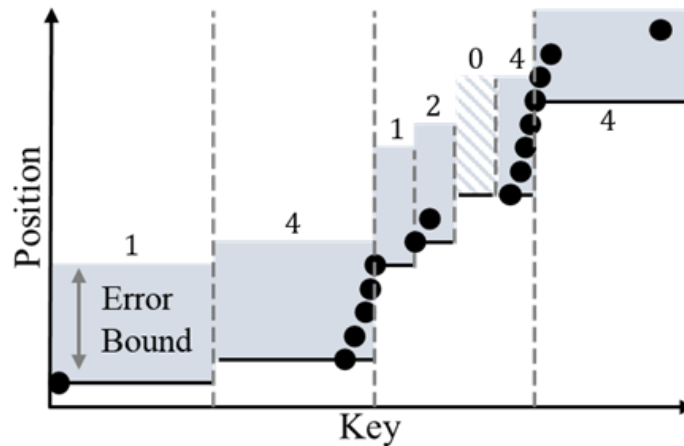


3. Design

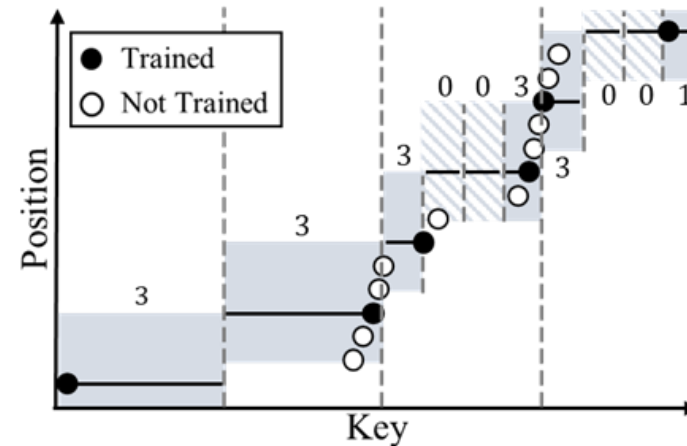
2. Sample Learning Algorithm

- Sample EB-Histogram

- Train **all** keys with the error-bound $\varepsilon \rightarrow k \in [p, p + \varepsilon]$
- Train the **sample** I^{th} keys with smaller error-bound $\delta (= \varepsilon - I + 1) \rightarrow k \in [p - I + 1, p + \delta]$
 - Preserve correction length ($\varepsilon + 1 = \delta + I$)



(a) EB-Histogram ($I = 1, \varepsilon = 5$)

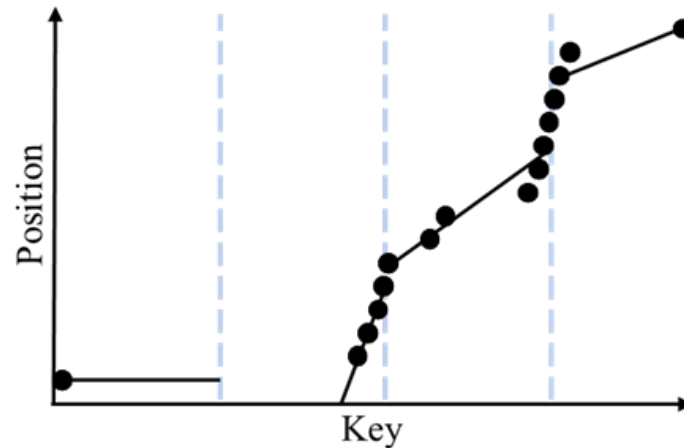


(b) Sample EB-Histogram ($I = 1, \delta = 3, \varepsilon = 5$)

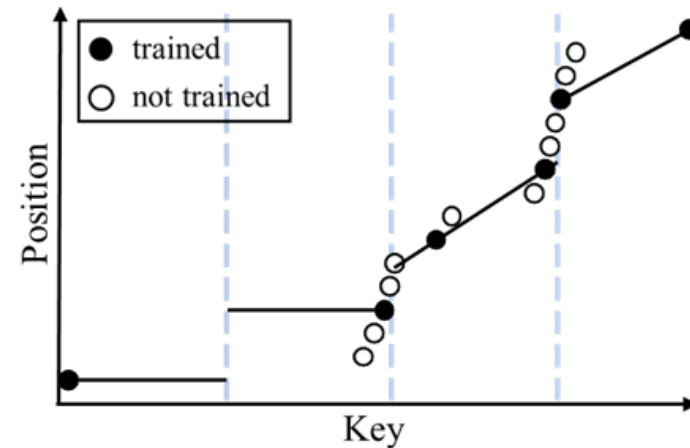
3. Design

2. Sample Learning Algorithm

- Simple Linear Regression
 - The model itself **cannot guarantee** the error-bound regardless of sampling
 - To guarantee error bounds, measuring the error of all data causes **complete traversal**
 - Train sample I^{th} keys \rightarrow Accuracy can decrease but the error-bound property doesn't change



(a) Simple Linear Regression ($I = 1$)



(b) Simple Linear Regression ($I = 3$)

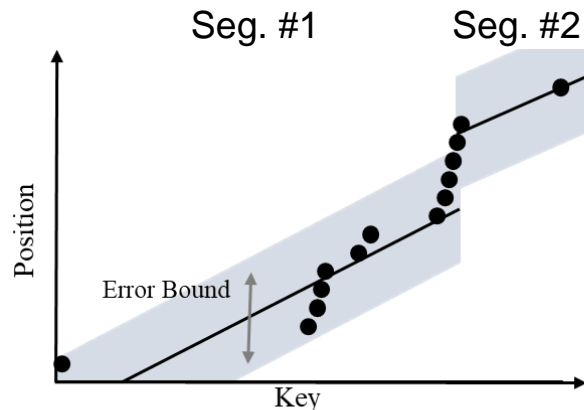
3. Design

3. Internal Changes due to Sampling

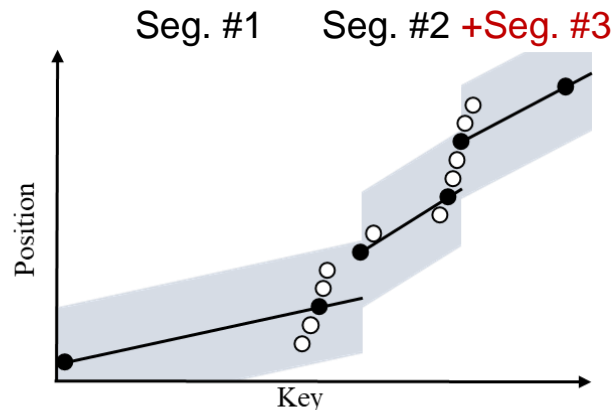
- Depend on segmentation manner

1) **Dynamic** segmentation (EB-PLA, EB-Histogram)

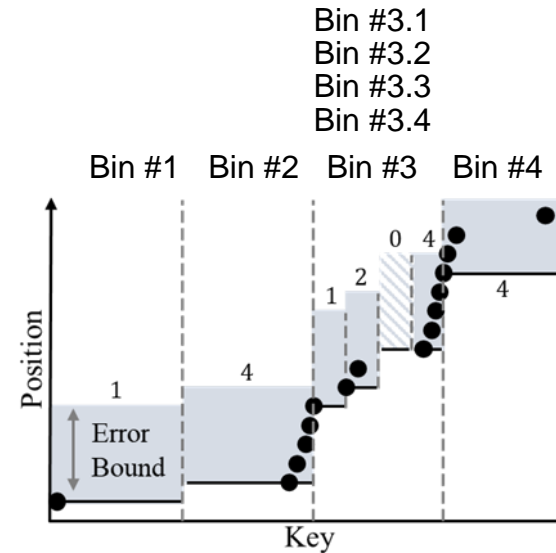
- Definition: Dynamically segment key ranges according to the distribution
- Trade-off: Decrease build time but aggressive sampling can increase **# of segments (bins)**



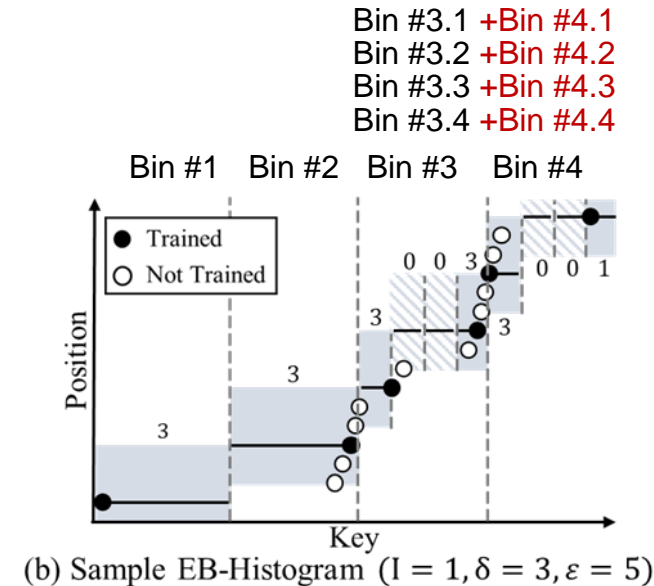
(a) EB-PLA ($I = 1, \varepsilon = 3$)



(b) Sample EB-PLA ($I = 1, \delta = 1, \varepsilon = 3$)



(a) EB-Histogram ($I = 1, \varepsilon = 5$)



(b) Sample EB-Histogram ($I = 1, \delta = 3, \varepsilon = 5$)

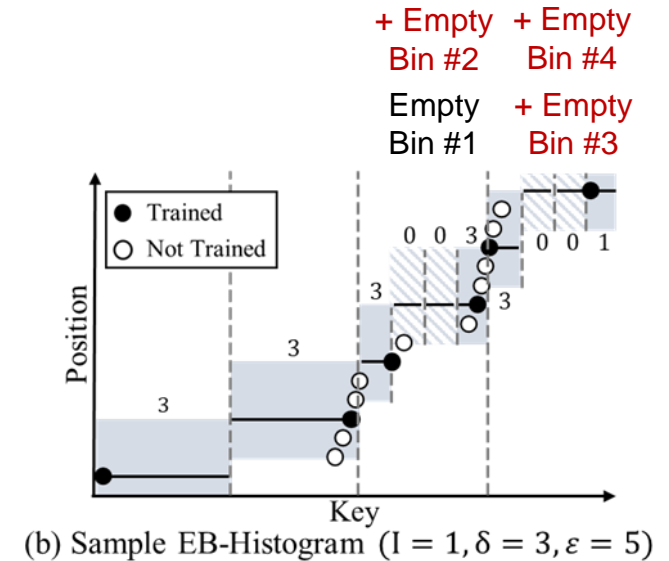
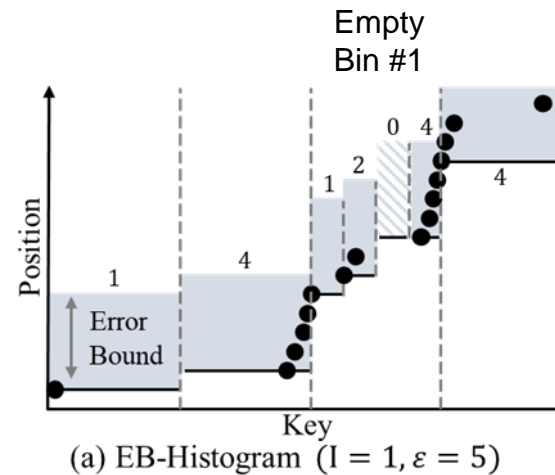
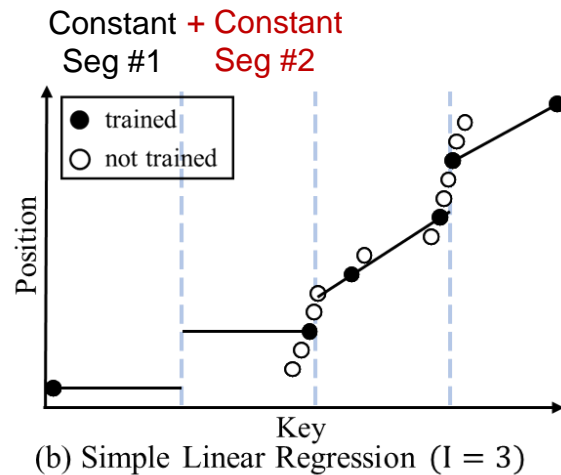
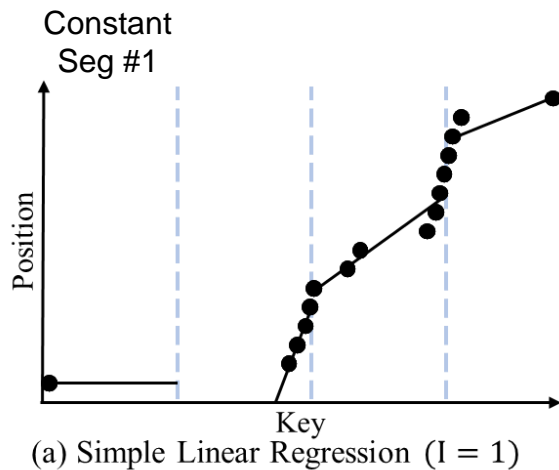
3. Design

3. Internal Changes due to Sampling

- Depend on segmentation manner

2) Fixed segmentation (Simple Linear Regression, EB-Histogram)

- Definition: Segment key ranges into a fixed number of segments
- Trade-off: Decrease build time but aggressive sampling can increase # of underfitting segments



4. Evaluation Setup



BASIL (Benchmark of Sampling Applied Learned Indexes)

- Applied sampling to 7 indexes, prefixed with “s”
 - 3 Learned, 2 Histogram, 3 Tree-based indexes

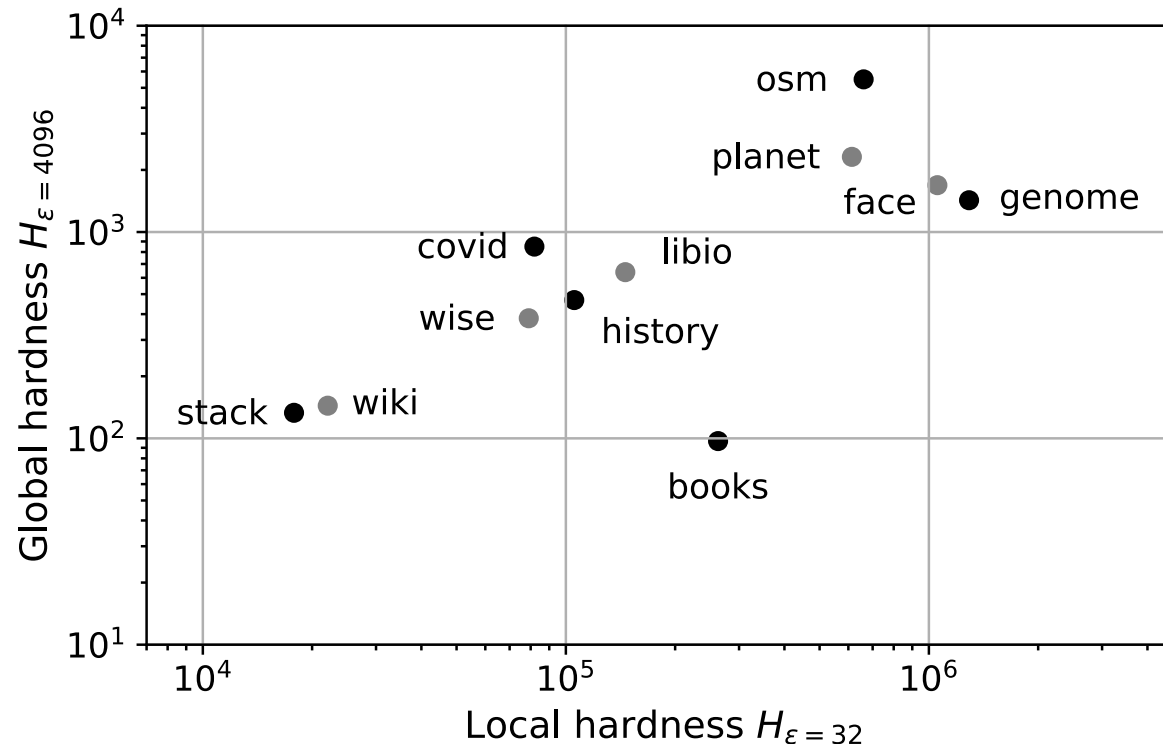
Type	Index	Internal Model	Correction Search
Learned	sRMI	Simple Linear Regression	Exponential Search
Learned	sPGM / sRS	Sample EB-PLA	Binary Search
Histogram	sCHT	Sample EB-Histogram (Equal-width)	
Histogram	sRT	Sample Histogram (Equal-width)	
Tree-based	sART / sB+-Tree/ sIB-Tree	-	

4. Evaluation Setup



BASIL (Benchmark of Sampling Applied Learned Indexes)

- Datasets: 6 representative datasets with 200 million key-value pairs
- Workload: Lookup uniform random 10 million keys from the dataset.
- Environment: Intel(R) Xeon(R) Gold 6338 CPU 2.00 GHz, 48 MB L3 cache with 512 GB of main memory



5. Evaluation

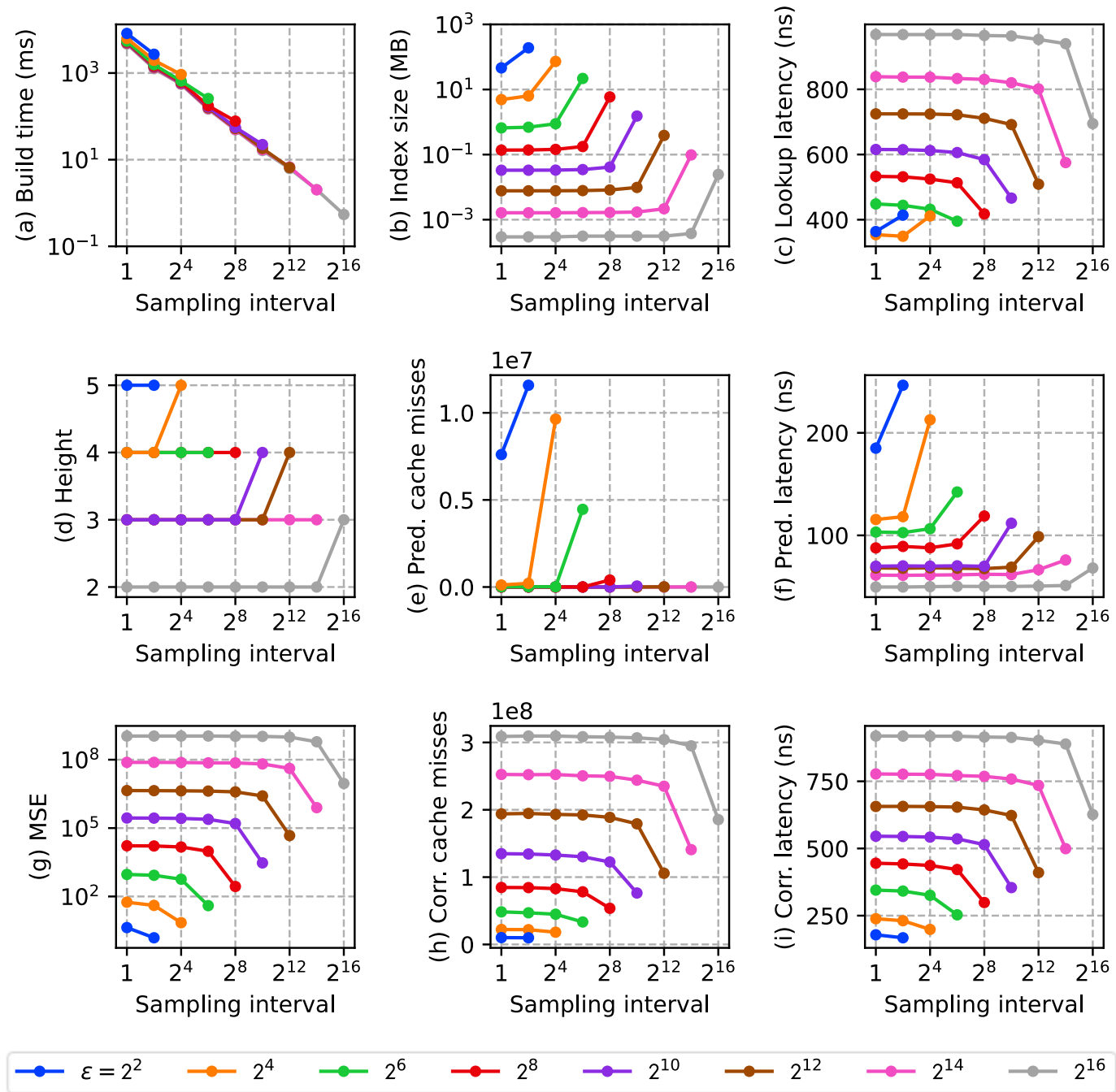
1. Sampling Trade-offs

Index

- sPGM

Metrics

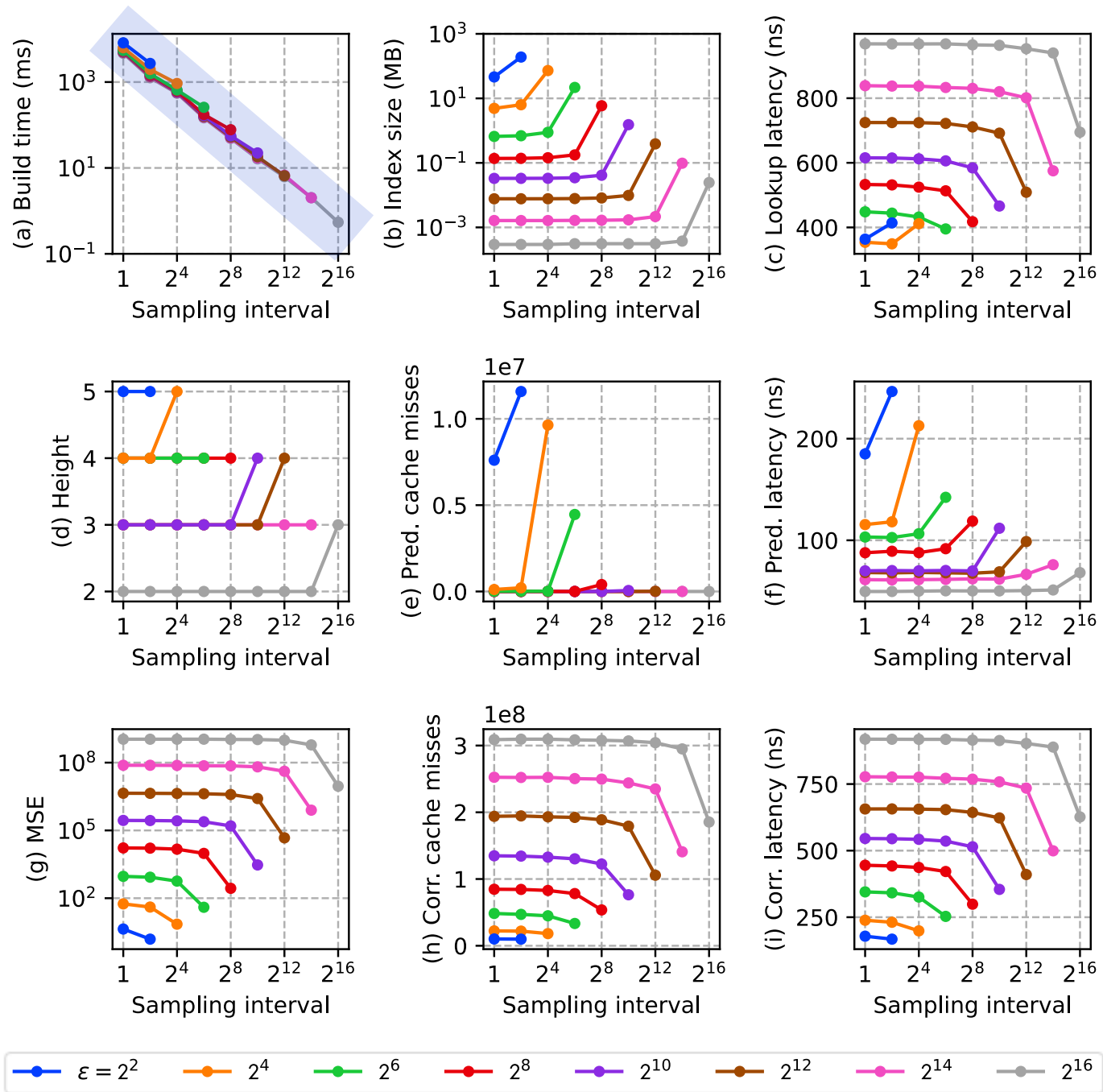
- Index:** (a) Build Time, (b) Size, (c) Latency, (f) Pred. latency, (i) Corr. latency
- Model:** (d) Height, (g) MSE (Accuracy)
- Micro-architecture:** (e) Pred. Cache Miss, (f) Corr. Cache Miss



5. Evaluation

1. Sampling Trade-offs

- When sampling interval (I) increases, **(a) build time decreases** by order of magnitude



5. Evaluation

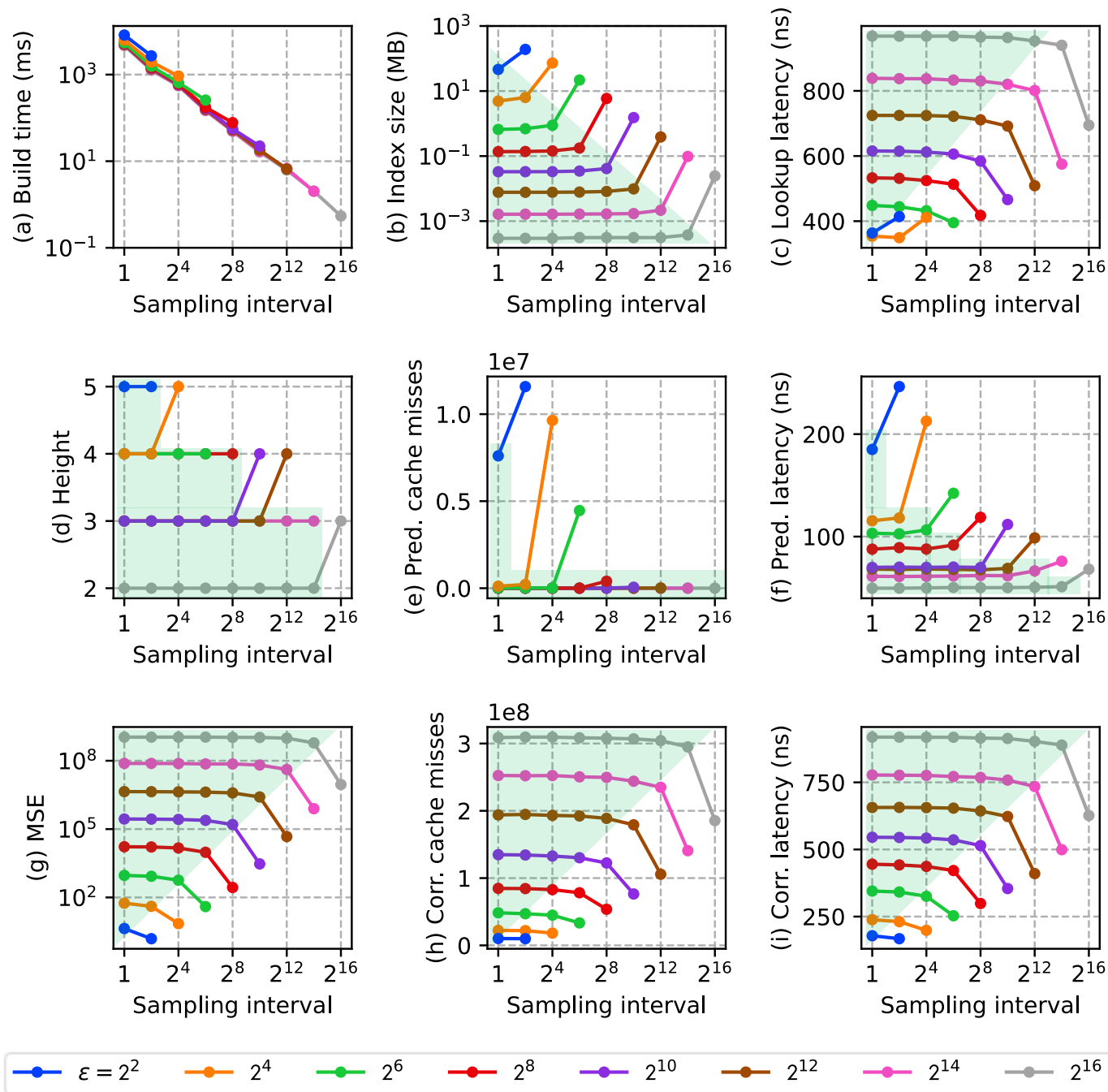
1. Sampling Trade-offs

- Each error-bound has a **threshold interval (I_{TH})**

- mostly $\varepsilon = I^{TH}$

- Until I_{TH} ,

(b-i) the rest of metrics remain consistent



5. Evaluation

1. Sampling Trade-offs

■ **After I_{TH} ,**

of linear segments \uparrow

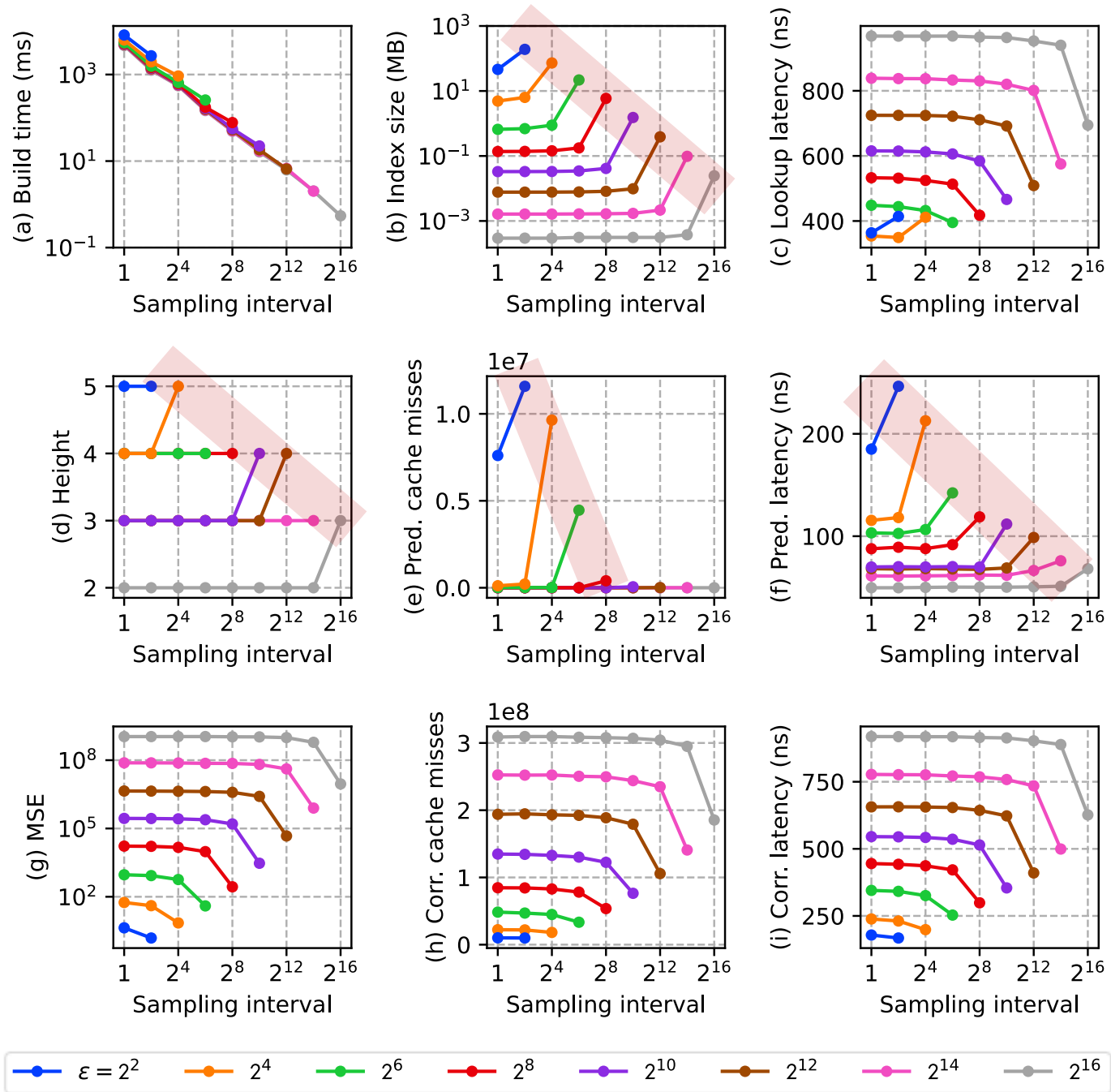
→ (b) Size \uparrow

(d) Height \uparrow

→ (e) Pred. cache miss \uparrow ,

(f) Pred. latency \uparrow

Dataset: History, Error bound ($\varepsilon \in [2^2, 2^{16}]$), Sampling interval ($I \in [2^0, \varepsilon (\leq 2^{16})]$)



5. Evaluation

1. Sampling Trade-offs

■ After I_{TH} ,

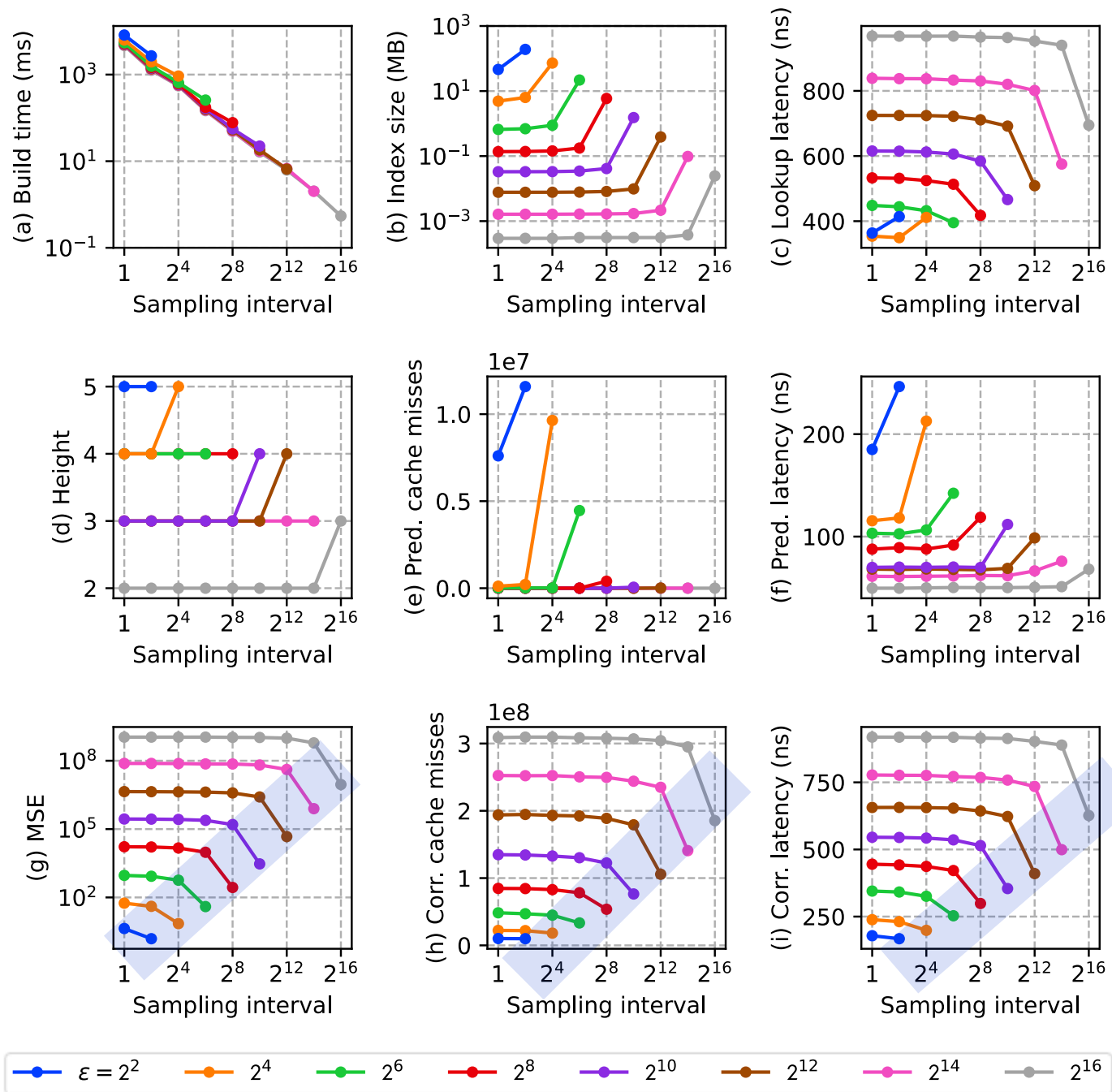
of linear segments \uparrow

→ (g) MSE \downarrow

→ (h) Corr. cache miss \downarrow

(i) Corr. latency \downarrow

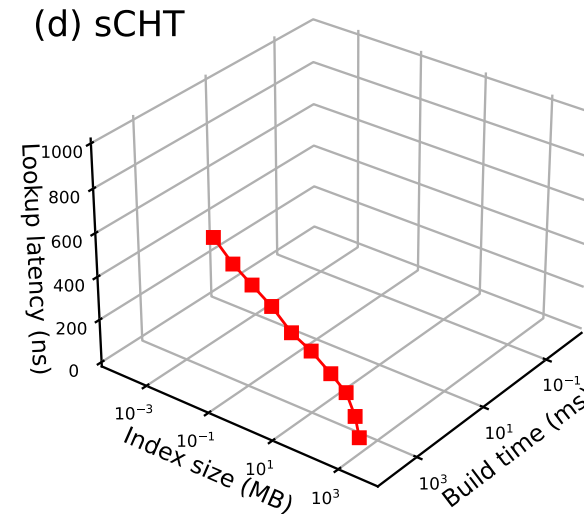
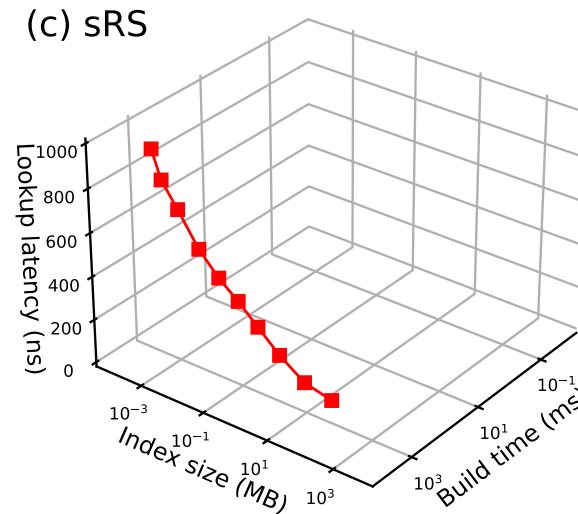
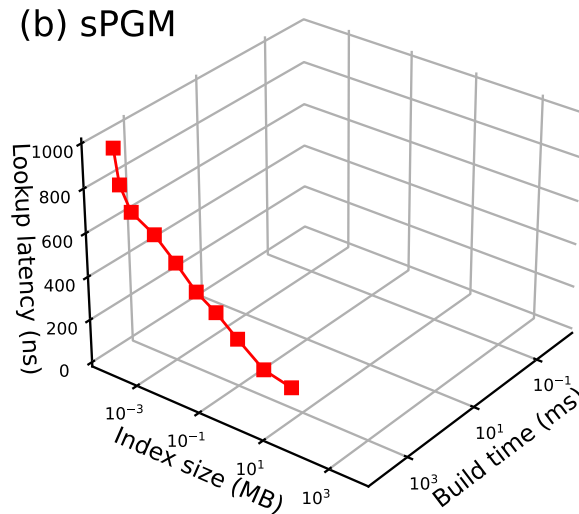
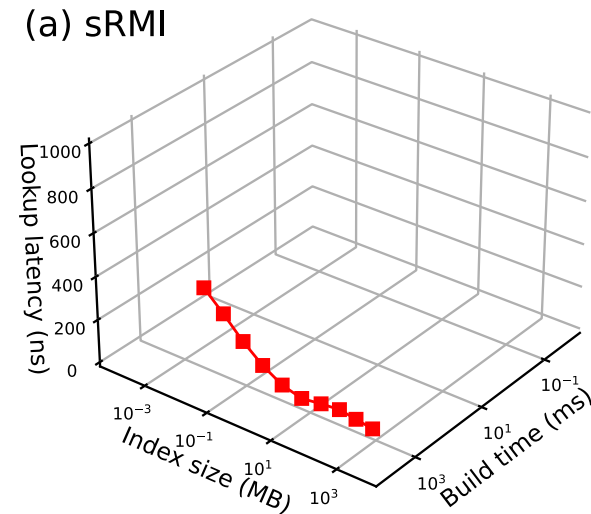
Dataset: History, Error bound ($\varepsilon \in [2^2, 2^{16}]$), Sampling interval ($I \in [2^0, \varepsilon (\leq 2^{16})]$)



5. Evaluation

2. Design Space of Learned Indexes

- **Absence** of trade-offs between build time, index size, and lookup latency
 - Incurs significant build times regardless of size and lookup

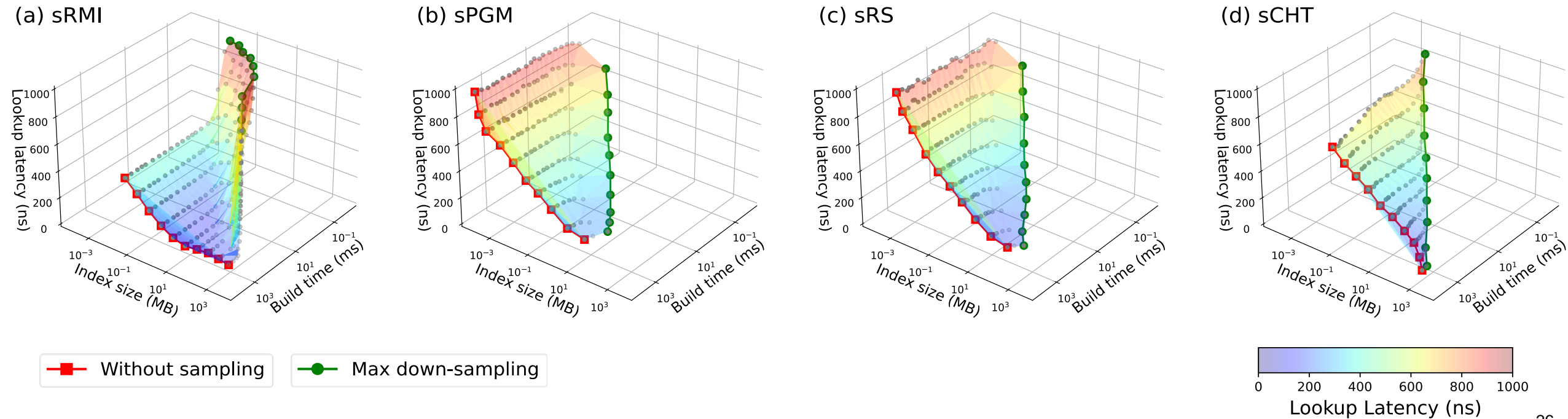


—■— Without sampling

5. Evaluation

2. Design Space of Learned Indexes

- Sampling **introduces** trade-offs between build-time, size, and lookup latency
 - Broadens** design space of learned indexes from 2D to 3D

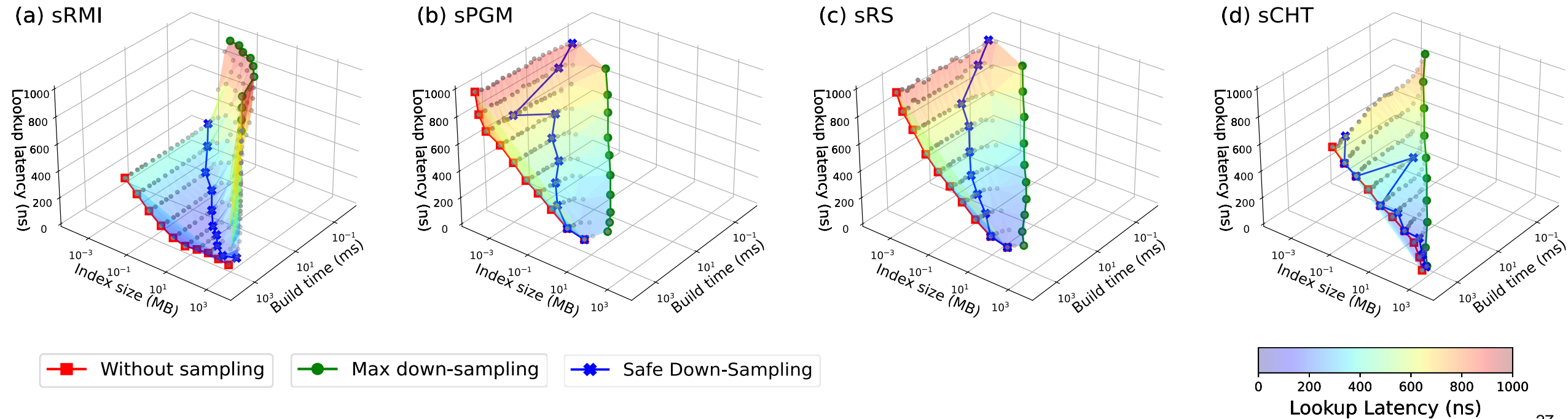


5. Evaluation

■ 3. Build Speed-up

- Question: How much can sampling reduce build time without significantly degrading index performance?

➤ **Safe** down-sampling where size and lookup latency increase by **less than 5%**

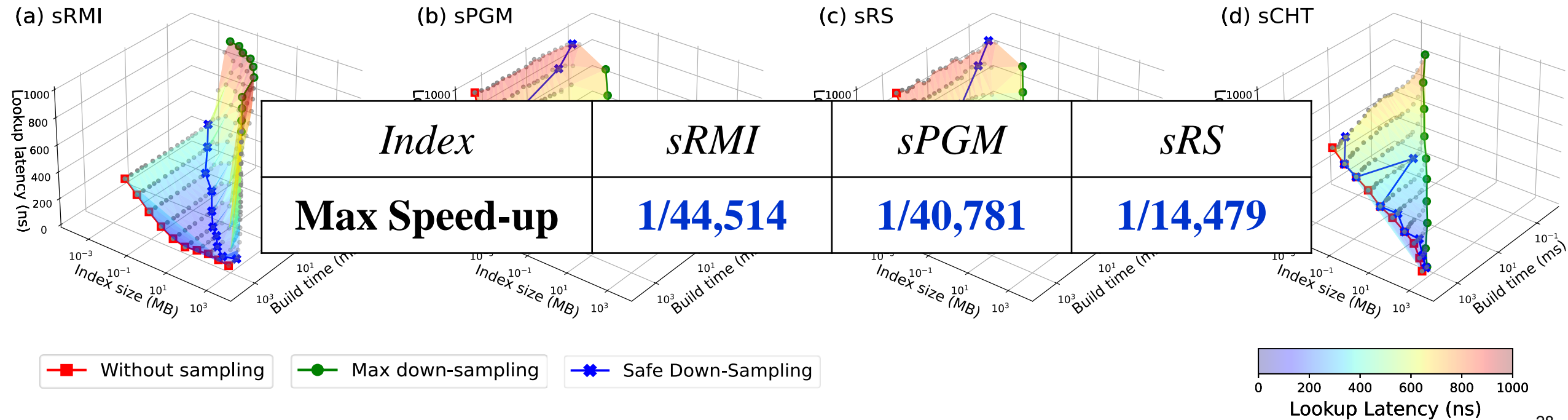


5. Evaluation

■ 3. Build Speed-up

- Question: How much can sampling reduce build time without significantly degrading index performance?

➤ **Safe** down-sampling where size and lookup latency increase by **less than 5%**



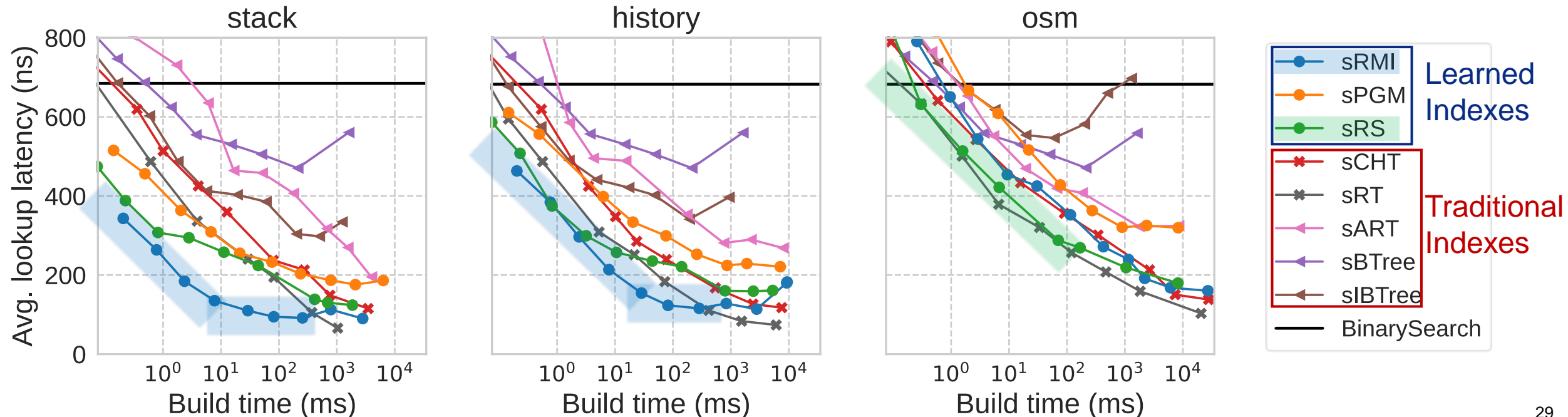
5. Evaluation

■ 4. Pareto Optimal Analysis

- Question: Can learned indexes be built more efficiently in terms of build time and lookup latency than traditional indexes through sampling?

- **Pareto optimal** (build-efficient) in terms of build time and **average** lookup latency

- no alternative that has both shorter build time and lower average latency



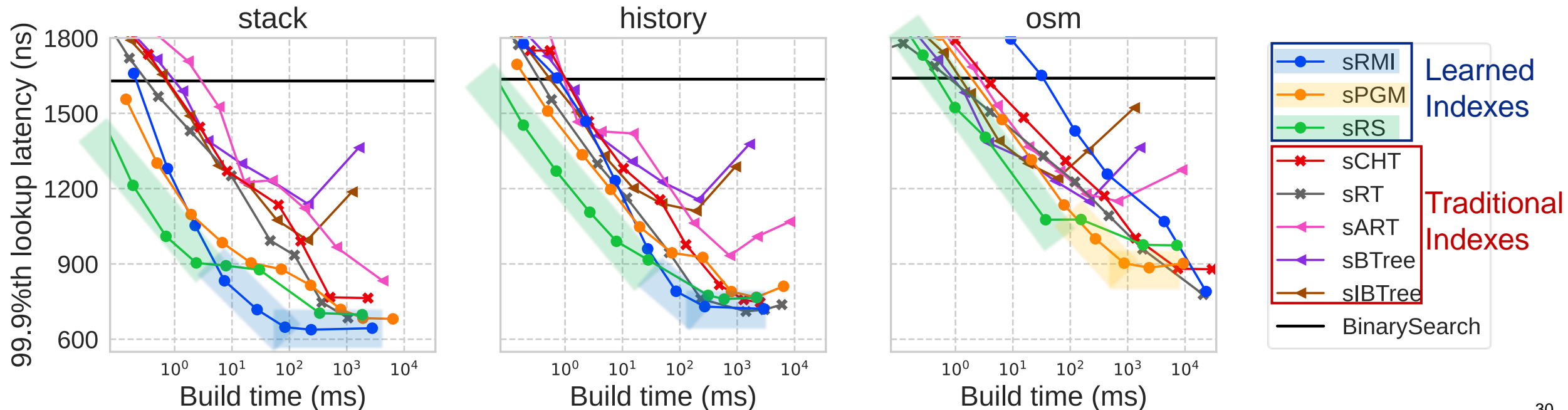
5. Evaluation

■ 4. Pareto Optimal Analysis


- Question: Can learned indexes be built more efficiently in terms of build time and lookup latency than traditional indexes through sampling?

- **Pareto optimal** (build-efficient) in terms of build time and **tail** lookup latency

➤ no alternative that has both shorter build time and lower tail latency



6. Conclusion

1. Learned indexes are space-efficient, but long build time make them impractical.
2. Sampling has 3 challenges: 1) losing the error-bound property, 2) absence of benchmark, and 3) complex sampling trade-offs.
3. We propose 1) novel sample learning algorithms that preserve the error-bound, 2) new benchmark,  BASIL, and 3) an analysis of sampling trade-offs.
4. We show that sampling can 1) expand the design space, 2) reduce build time without significant performance loss, and 3) build learned indexes efficiently.

Thank you