Can Learned Indexes be Built Efficiently? A Deep Dive into Sampling Trade-offs

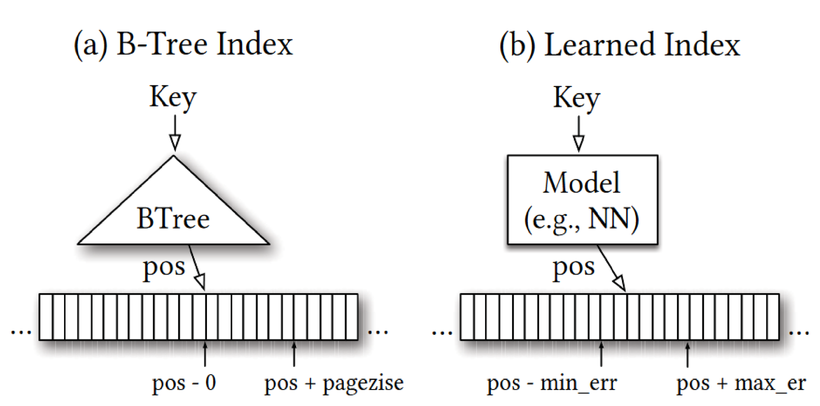
Minguk Choi, Seehwan Yoo, and Jongmoo Choi
Dankook University, South Korea
{mgchoi, seehwan.yoo, jmchoi}@dankook.ac.kr

Seeking a Ph.D. Position for 2025

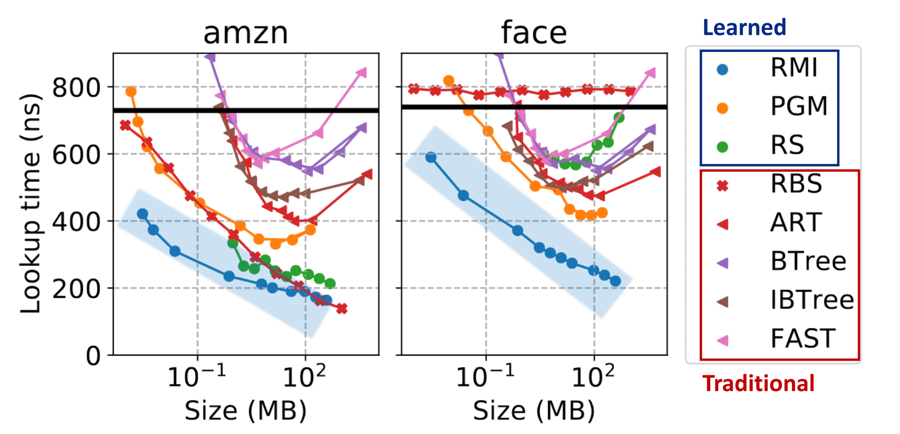
Santiago Chile
SIGMOD PODS 2024

Background

- **Learned Index Structure**
  - Index structure employs machine learning techniques
  - View the index as a model that predicts the position of a key
- **Performance of Learned Index : Space-efficient**
  - Pareto optimal for index size and lookup latency in read-only
    - No alternative exists that has both a smaller size and lower latency



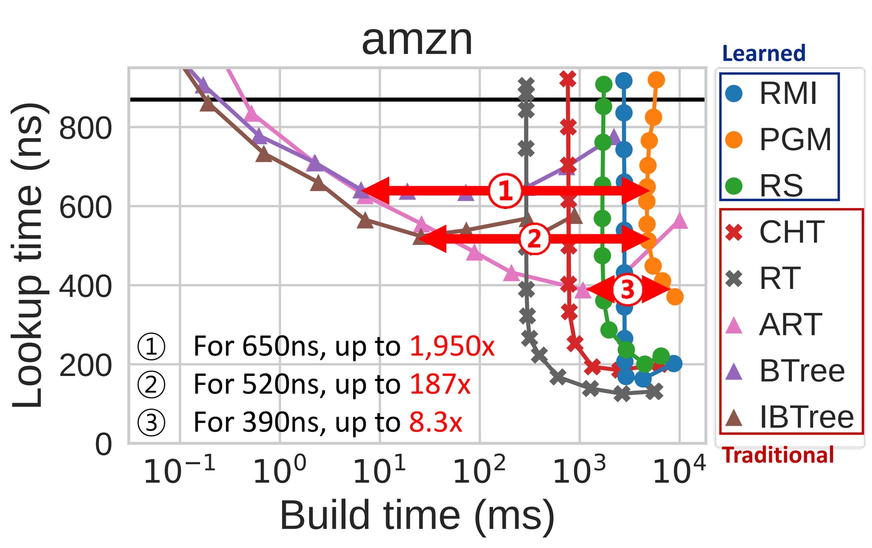
(The Case for Learned Index Structures, SIGMOD '18)



(Benchmarking Learned Indexes, VLDB '20)

Motivation

- **Long Index Build Time**
  - Up to about 2,000x longer than traditional indexes
  - But there are still application where index build time is crucial (e.g., LSM-tree)



- **Why Building the Learned Index is Slow?**

$$Index\ build\ time = 1)\ Number\ of\ elements \times 2)\ Per-element\ overhead$$

1) Complete traversal and training
2) Higher per-element training overhead
   - Light-weight training model : RadixSpline (aiDM`20), Bourbon (OSDI`20)
     - But it's still longer than traditional indexes

- **This study began with this question ...**

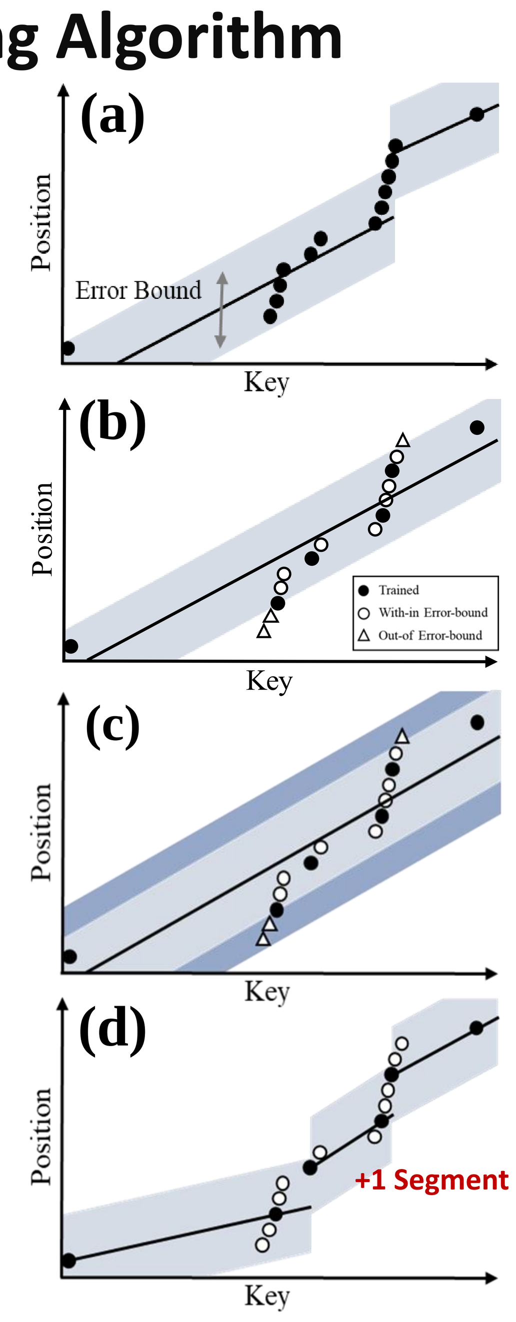
Since a learned index uses a model,
Can't it learn efficiently even with less data?

Design

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

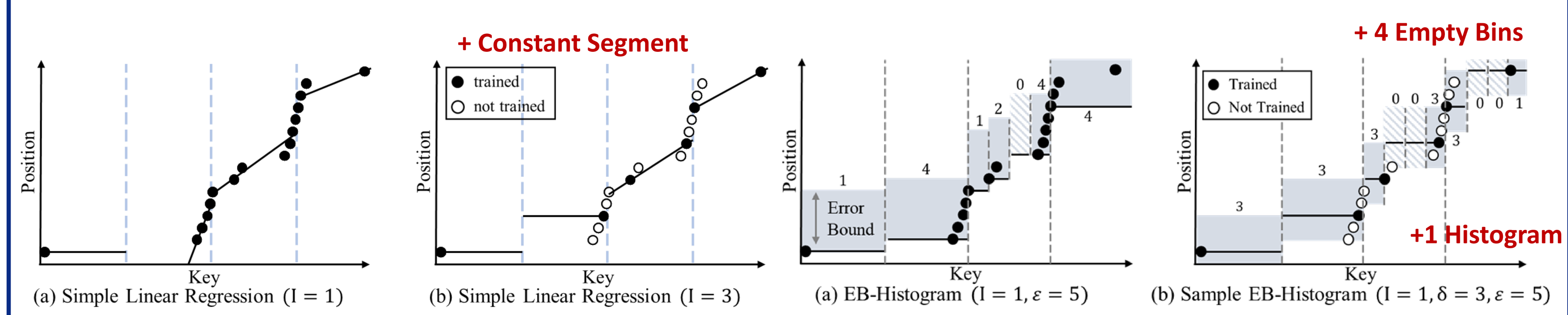
1. Error-bound Preserving Sample Learning Algorithm

- EB-PLA (Error-bounded Piece-wise Linear Approximation) Model
  - (a) Without sampling, model trains all keys with error-bound $\varepsilon \rightarrow \forall k, Error(k) \leq \varepsilon$
  - (b) With sampling, model trains sample I^{th} keys with error-bound $\varepsilon \nrightarrow \forall k, Error(k) \leq \varepsilon$
- **Sample EB-PLA** Algorithm
  - (c) Refine error-bound due to sampling loss $\rightarrow \forall k, Error(k) \leq \varepsilon' (= \varepsilon + I - 1)$
    - Preserve error-bound property
  - (d) Replace sample learning error-bound to $\delta\ (= \varepsilon - I + 1)$ for desired error-bound (ε)
    - Preserve error-bound (ε) by learning less data with smaller & stricter error-bound (δ)
- **Sample EB-Histogram**
- PLR with Simple Linear Regression



2. Internal Changes due to Sampling

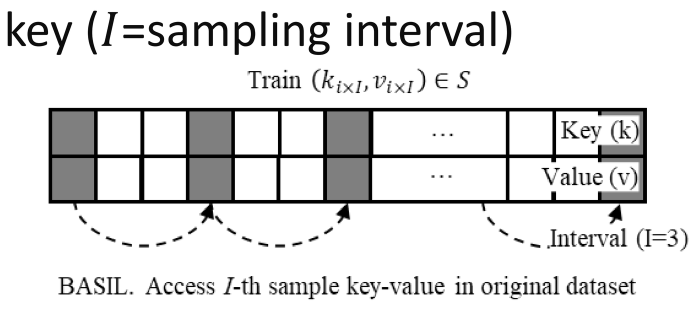
1) Dynamic Segmentation (Key range of each segments are different)
   - Aggressive sampling can increase number of segments
2) Fixed Segmentation (Key range of each segments are equal)
   - Aggressive sampling can increase number of under-fitting segments



3. Unified Sampling Algorithm& Implementation

BASIL (Benchmark of Sampling Applied Learned Indexes)

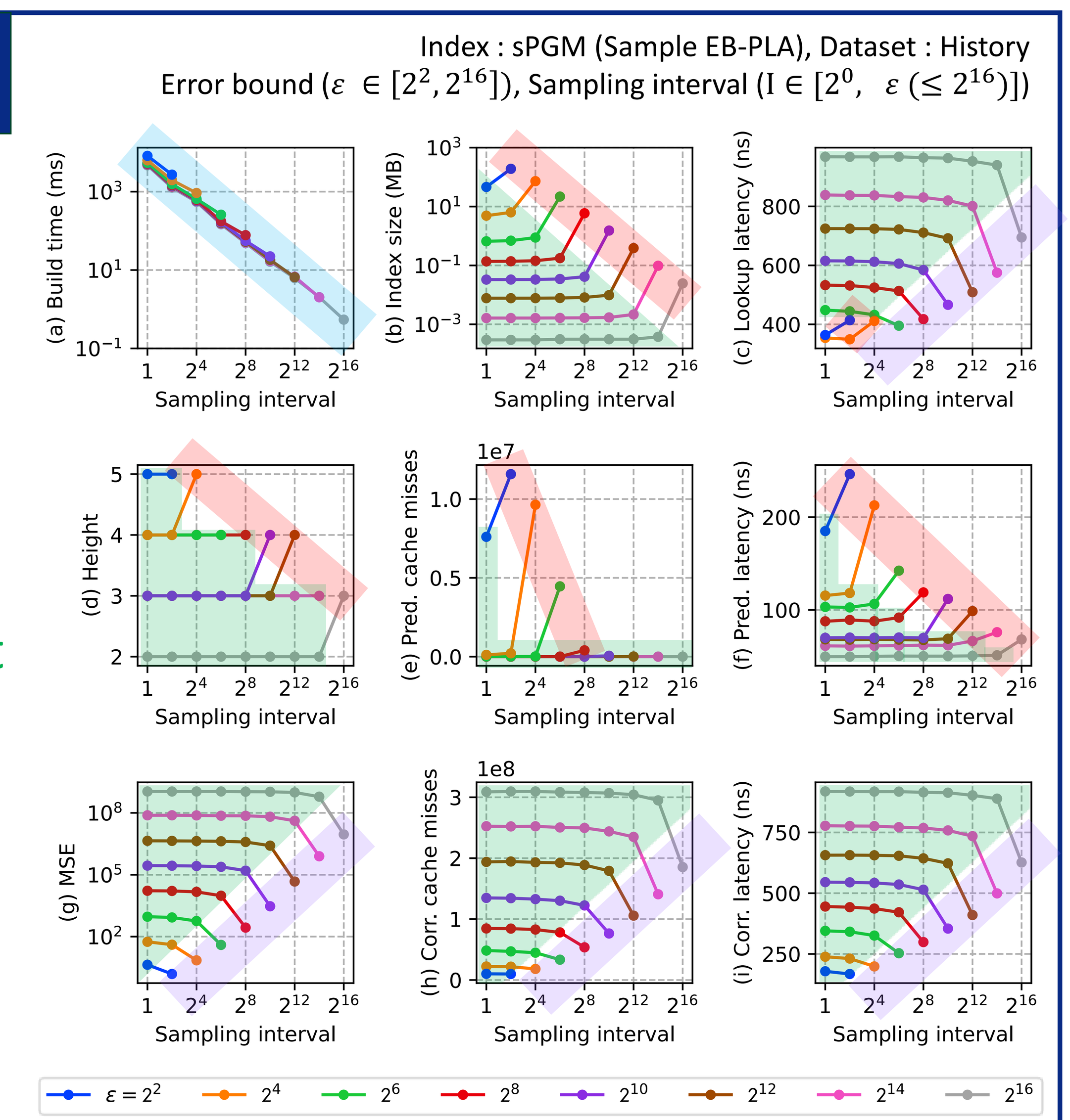
1) Unified Sampling Algorithm : Systematic Sampling
   - Extract every I^{th} key form first to last key (I=sampling interval)
2) Unified Sampling Implementation
   - Index access and train only sample key-value data from entire dataset



Evaluation

1. Sampling Trade-offs

- Sampling interval (I)↑ → (a) build time ↓
- Each error-bound (ε) has **threshold interval (I^{TH})**
- Until I^{TH}, (b-i) rest of metrics remain consistent
- After I^{TH}, # of segments ↑ → (b) Size ↑, (d) Height ↑ → (e) Pred. cache miss ↑, (f) Pred. latency ↑
- After I^{TH}, # of segments ↑ → (g) MSE ↓ → (h) Corr. cache miss ↓, (i) Corr. latency ↓

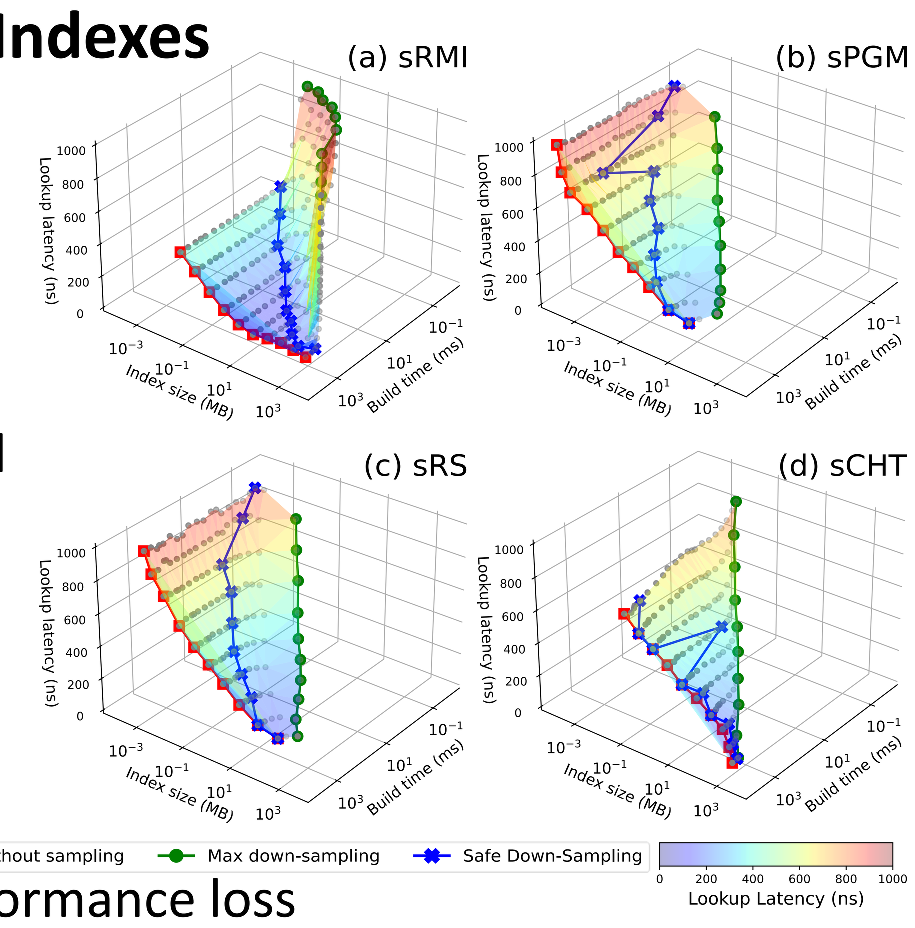


2. Design Space of Learned Indexes

- Without sampling, absence of trade-offs between build, size, and lookup
- Sampling introduce trade-offs between build, size, and lookup
- **Broaden** design space of learned indexes from 2D to 3D

3. Build Speed-up

- Explore **Safe down-sampling**, where size & lookup latency increased by less than 5%
- Max build speedup without performance loss
  - sRMI : 1/44,514, sPGM : 1/40,781, sRS : 1/14,479



4. Pareto Analysis

- **Can learned indexes be built efficiently** (in terms of build time and lookup latency) than traditional indexes through sampling)?
- To best of our knowledge, it is first to show that learned indexes are also **Pareto optimal** in build time and (avg. & tail) lookup latency through sampling