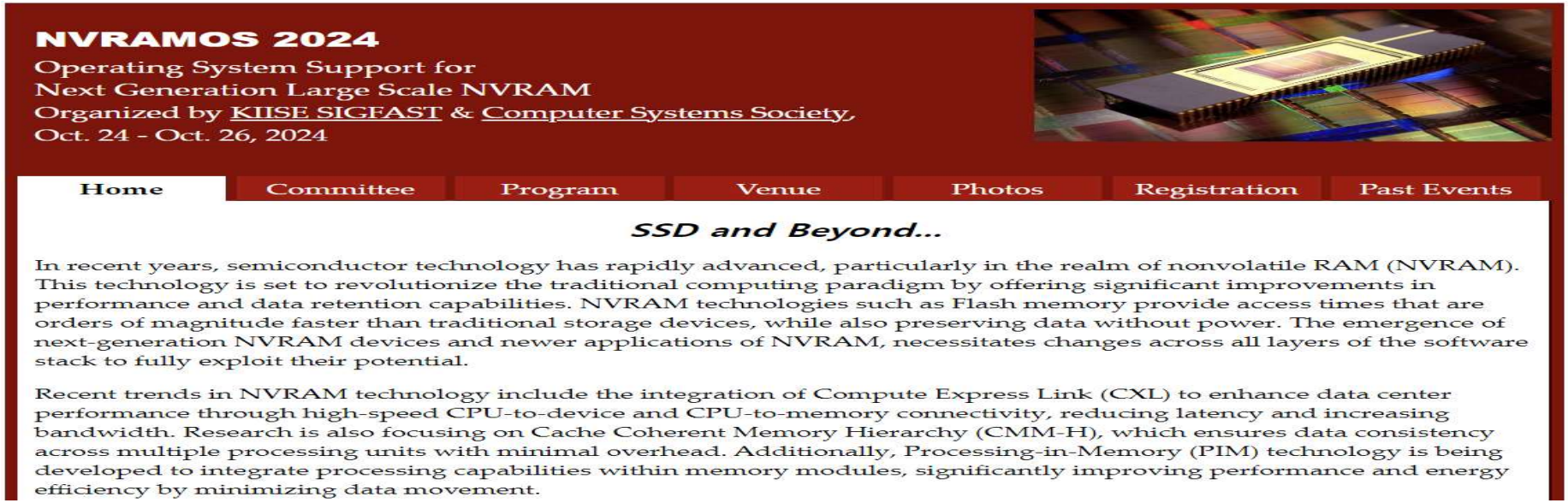
Can Learned Indexes be Built Efficiently? A Deep Dive into Sampling Trade-offs

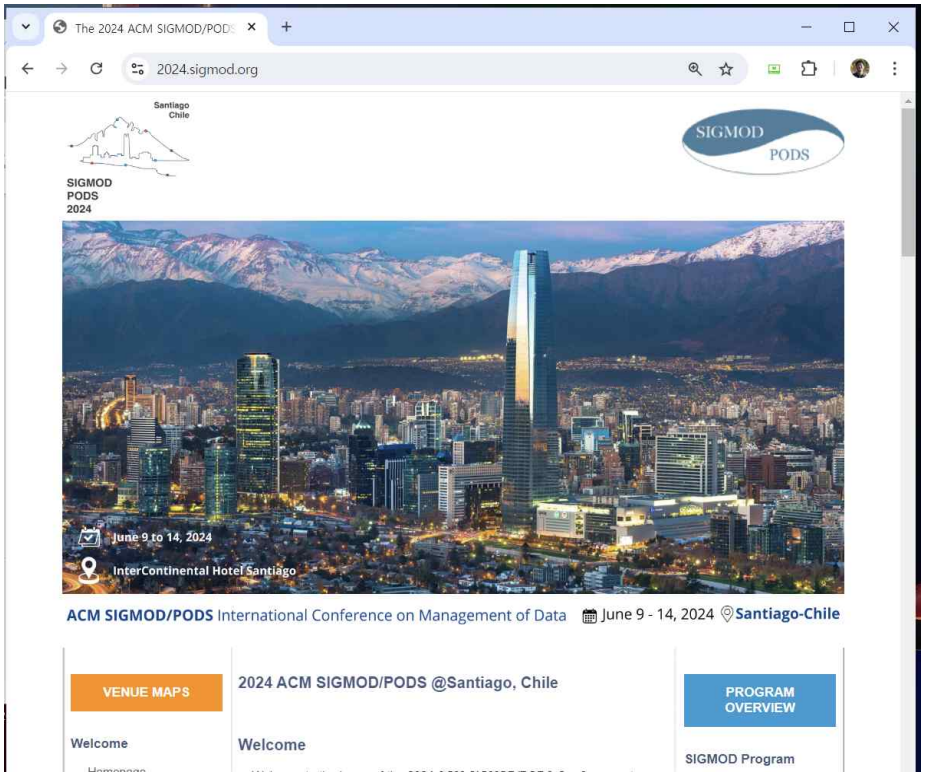


October 24, 2024

Jongmoo Choi
http://embedded.dankook.ac.kr/~choijm

Prologue

- Appeared
  - ✓ ACM SIGMOD'24
- Contributors
  - ✓ Minguk Choi, Seehwan Yoo, Jongmoo Choi



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

MINGUK CHOI, Dankook University, South Korea
SEEHWAN YOO*, Dankook University, South Korea
JONGMOO CHOI*, Dankook University, South Korea

By embedding the distribution of keys in indexing structure, learned indexes can minimize the index size and maximize the lookup performance. Yet, one of the problems in the present learned index is the long index-building time. The conventional learned index requires a complete traversal of the entire dataset, which makes it less practical than traditional index. This paper challenges the efficiency of build time to make the learned index practical.

Our approach for a build time-efficient learned index is to employ sampled learning. In this paper, we present two error-bounded sampling schemes: Sample EB-PLA, and Sample EB-Histogram. Although sampling is a simple idea, there are several considerations to make it practical. For example, sampling interval, error-boundness, and index hyper-parameters are inter-related each other, presenting complicated trade-offs between build-time, index size, accuracy and lookup latency.

Throughout the extensive experiments over six real-world datasets, we show that the index-building time can be efficiently reduced over an order of magnitude by our sampling schemes. The results reveal that the sampling expands the design space of learned indexes, including the build-time as well as lookup performance and index size. Our Pareto analysis shows that a learned index can be built more efficiently than a traditional index through sampling.

CCS Concepts: • **Information systems** → **Data access methods**; • **Computing methodologies** → **Learning linear models**.

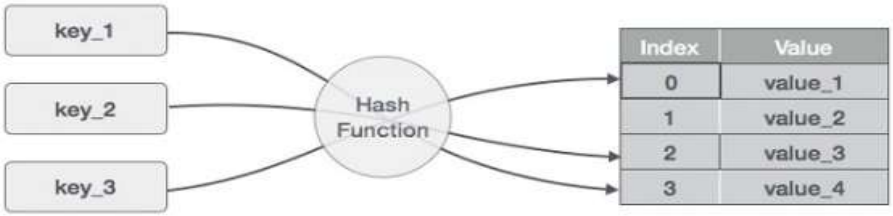
ACM Reference Format:
Minguk Choi, Seehwan Yoo, and Jongmoo Choi. 2024. Can Learned Indexes be Built Efficiently? A Deep Dive into Sampling Trade-offs. *Proc. ACM Manag. Data* 2, 3 (SIGMOD), Article 116 (June 2024), 25 pages. https://doi.org/10.1145/3654919

Contents

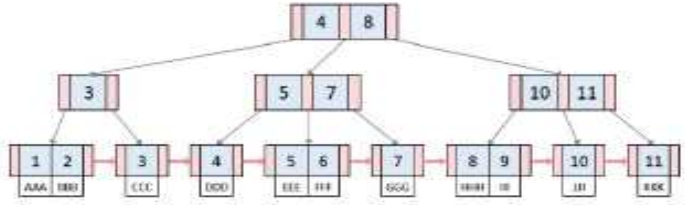
- Introduction
  - ✓ What is Index?
  - ✓ What is Learned index?
- Motivation
  - ✓ Issues on Learned index
  - ✓ Space vs Lookup vs Build
- Challenges and Our solutions
  - ✓ Preserving error-bound property ➔ Systematic sampling and EB-PLA
  - ✓ Complex trade-offs ➔ Applying sampling into diverse learned indexes
  - ✓ Fair comparison ➔ BASIL
- Evaluation
  - ✓ Trade-offs analysis
  - ✓ Explore new design space
- Conclusion
- Appendix
  - ✓ Integrate Learned index with Key-Value Store

Introduction (1/6)

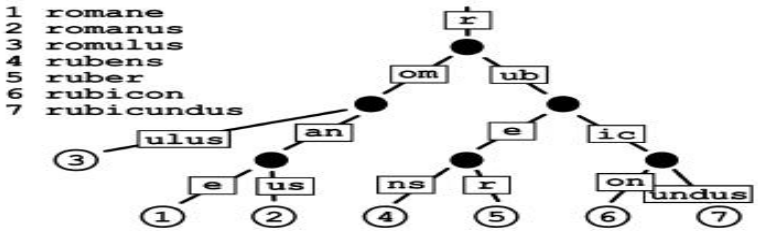
- What is Index?
  - ✓ Definition from Wikipedia
    - An index is used to optimize performance in finding relevant documents for a search query (without an index, we need to scan every document)
    - A data structure for improving the data retrieval speed
  - ✓ Types
    - 1) Hash-based, 2) Tree-based, 3) Trie-based, 4) List-based, …



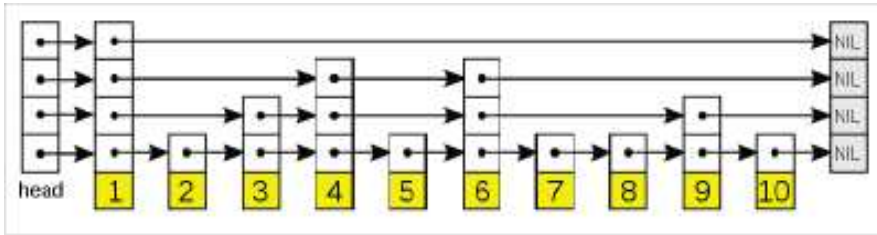
(Hash)



(B+-tree)



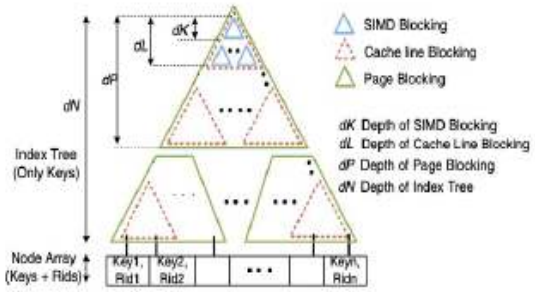
(Radix tree or compressed trie)



(Skiplist)

Introduction (2/6)

- Index optimization
  - ✓ Micro-architecture awareness
    - FAST: pack parent and child nodes into a fat node ➔ SIMD + Cache friendly + No pointer operations (better pipeline effect)
  - ✓ Flexibility
    - ART (Adaptive Radix Tree): different number of items in a node ➔ space efficiency and cache friendly
  - ✓ Hybrid
    - S3: Skiplist + FAST and Semi-ordered ➔ accelerating top list search in Skiplist
  - ✓ For disaggregated system
    - SMART: read delegation and write combining, hybrid concurrency control ➔ reduce read and write amplification between nodes



(FAST, SIGMOD'10)

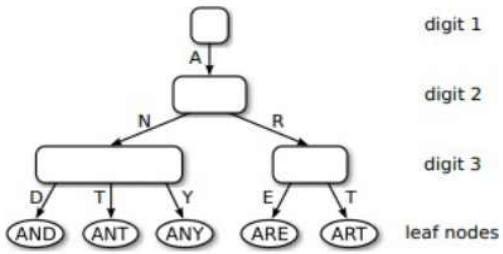
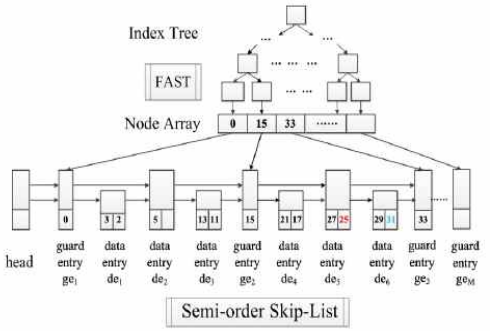
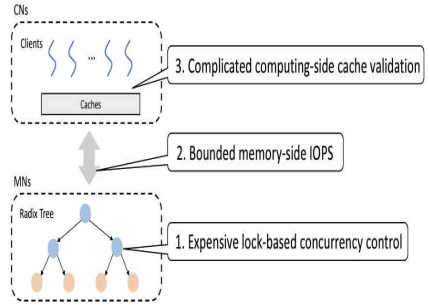


Fig. 1. Adaptively sized nodes in our radix tree.

(ART, ICDE'13)



(S3, VLDB'19)



(SMART, OSDI'23)

Introduction (3/6)

- What is Learned Index?
  - ✓ Proposed by Tim Kraska (MIT@CSAIL) in 2018
    - The Case for Learned Index Structures (SIGMOD '18)
      - Most influential database paper in 2018
  - ✓ Definition
    - Instead of data structure, use a model (ML) for indexing
    - Less memory footprint, faster lookup, …
  - ✓ A lot of model proposed recently

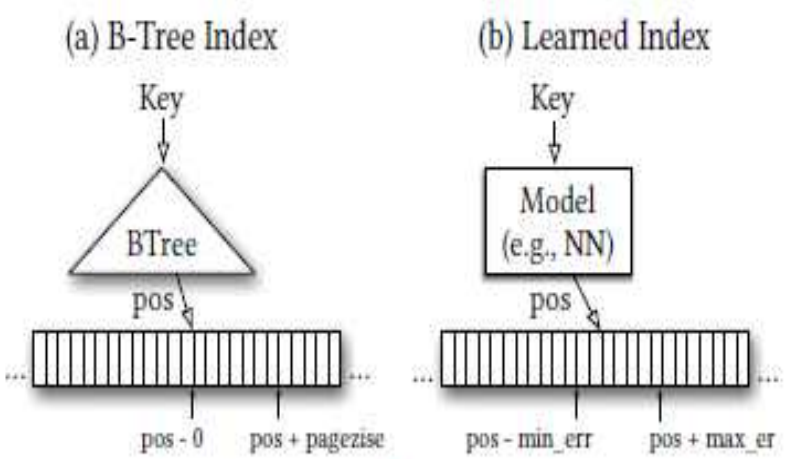


Figure 1: Why B-Trees are models

(The Learned Index, SIGMOD'18)

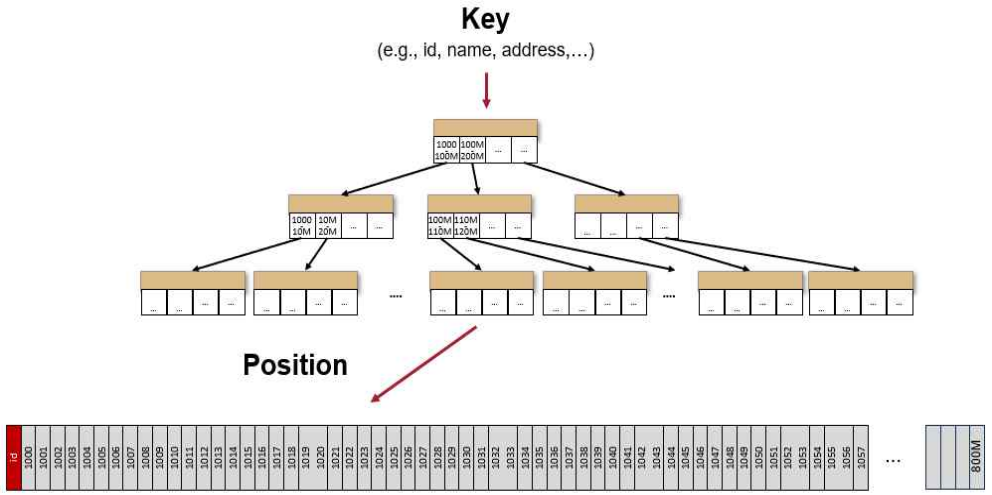
	Index	Insert		Lookup		Concurrency	Bulk Loading
		Insert Strategy	Structural Modification	Data Fitting Model	Position Search		
Learned	RMI [18]	No	No	Simple neural network	At leaf nodes	No	Top-down
	PLEX [41]	No	No	Non-linear model [4] Linear interpolation	At all nodes	No	Greedy split Bottom-up
	PGM [11]	No	No	Linear model	At all nodes	No	Greedy split Bottom-up
	DPGM (Dynamic PGM [11])	Delta-buffer	Buffer merge [35]	Linear model	At all nodes	No	Greedy split Bottom-up
	XIndex [43]	Delta-buffer	Buffer merge Error-based node split	RMI Piecewise linear regression	At all nodes	Temporary buffer	Even split Bottom-up
	FINEdex [24]	Delta-buffer	Fullness-based buffer train&merge	Piecewise linear regression	At all nodes	Pair-level buffer Buffer train&merge	Greedy split Bottom-up
	SIndex [46]	Delta-buffer	Buffer merge	(Piecewise) linear regression	At all nodes	Temporary buffer	Greedy split Bottom-up
	ALEX [7]	In-place	Fullness&cost based node expand/split/rebuild	Linear model	At leaf nodes	No	Cost-based split Top-down
	MAB+tree [1]	In-place	Fullness-based node split	Linear interpolation	At all nodes	No	Greedy split Bottom-up
	LIPP [49]	In-place	Conflict&fullness based subtree rebuild	Non-linear model	No	No	Conflict-based split Top-down
Traditional	FAST [14]	No	No	No	At all nodes	No	Bottom-up
	ART [20]	In-place	Fullness&prefix based node expand/split	No	At most nodes	No	Top-down
	B+tree [3]	In-place	Fullness-based node split	No	At all nodes	No	Bottom-up
	Wormhole [52]	In-place	Fullness-based node split	No	At all nodes	RCU [32] hash table	Bottom-up

(Comprehensive Eval., VLDB'23)

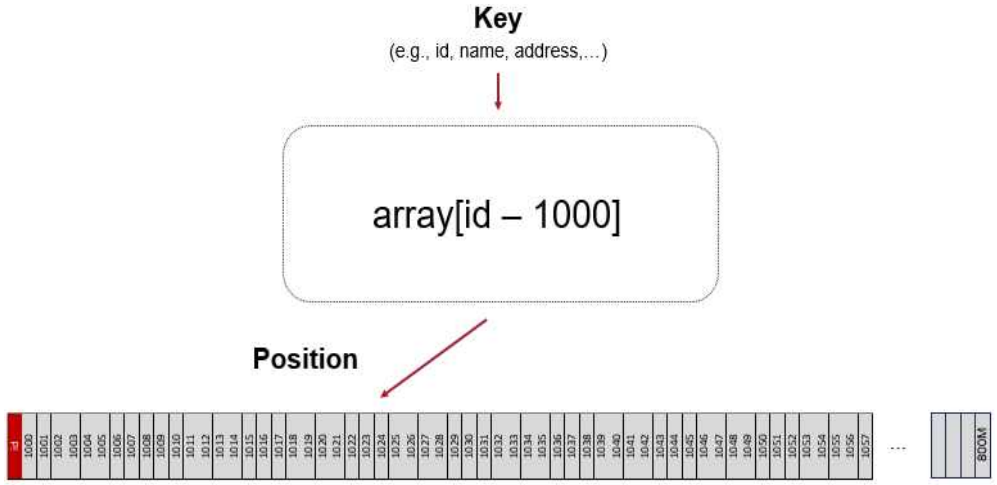
Introduction (4/6)

- Key Idea of Learned Index
  - ✓ Exploiting data distribution

id	date	first_name	last_name	email	address	zip	state	credit_card_nb	amount
1000	2017-01-01	Hobart	Spracklin	hspracklin0@dailymotion.com	20565 High Crossing Plaza	56372	Minnesota	4405-6975-7285-5160	$611.00
1001	2017-01-02	Billye	Binnion	bbinnion1@123-reg.co.uk	3698 Upham Point	20260	District of Columbia	3533-7150-7728-9850	$244.00
1002	2017-01-02	Johann	Brockley	jbrockley2@bizjournals.com	23844 Artisan Place	98516	Washington	67597-1193-7985-5100	$233.00
1003	2017-01-03	Artie	MacMenami	amacmenamin3@hao123.com	6276 Toban Trail	78759	Texas	3537-4829-6134-5000	$210.00
1004	2017-01-03	Delilah	O'Currigan	docurrigan4@chron.com	86016 New Castle Avenue	72199	Arkansas	3555-2017-2226-5780	$286.00
1005	2017-01-04	Gretta	Will	gwill5@yelp.com	0 Dottie Circle	68524	Nebraska	503844-1984-2085-5000	$870.00
1006	2017-01-04	Gordon	Kirsopp	gkirsopp6@utexas.edu	64060 Scott Park	20370	District of Columbia	633332-1895-2414-5000	$687.00
1007	2017-01-05	Bendick	Fagg	bfagg7@army.mil	94 Florence Hill	45440	Ohio	3528-9673-1815-8420	$733.00
1008	2017-01-05	Dimitry	Boyet	dboyet8@sakura.ne.jp	35886 Golf Plaza	30066	Georgia	3576-6991-4041-3170	$382.00
1009	2017-01-06	Ailsun	Beinke	abeinke9@si.edu	1 Badeau Place	46295	Indiana	56022-2011-8072-1400	$854.00
1010	2017-01-07	Lou	Hallows	lhallowsa@theguardian.com	1 Twin Pines Junction	91125	California	5602-2364-4079-0250	$150.00
1011	2017-01-09	Tiffani	Mathew	tmathewb@seattletimes.com	0456 Meadow Vale Lane	75260	Texas	6387-6943-8910-4580	$313.00
1012	2017-01-09	Perl	Bridie	pbridiec@hubpages.com	07 Bluestem Junction	33124	Florida	3539-8662-2397-5880	$558.00
1013	2017-01-09	Rosabelle	Blasik	rblasikd@delicious.com	7 Fairfield Pass	79699	Texas	5602-2297-6599-8560	$941.00
1014	2017-01-10	Meggi	Belamy	mbelamye@ask.com	0995 Manufacturers Street	10170	New York	3557-5094-7405-8340	$875.00
1015	2017-01-10	Tadio	Balderston	tbalderstonf@apache.org	80 Novick Road	75260	Texas	60485-3728-7119-9300	$954.00
1016	2017-01-11	Gianina	Oxteby	goxtebyg@google.pl	72674 Fuller Avenue	89505	Nevada	4-0415-9268-2397	$239.00
1017	2017-01-12	Brendan	Doody	bdoodyh@craigslist.org	87414 Golden Leaf Street	11480	New York	201-6348-4121-1314	$308.00
1018	2017-01-13	Conway	Coombs	ccoombsi@blogger.com	2810 Oakridge Park	32859	Florida	3529-1514-0357-9120	$ 60.00
1019	2017-01-14	Germaine	Bere	gberej@bravesites.com	82802 Oakridge Park	20041	District of Columbia	670961-0240-4054-9000	$ 95.00
1020	2017-01-15	Davide	Tolcharde	dtolchardek@redcross.org	89 Continental Avenue	79165	Texas	5018-7748-4325-9510	$137.00
1021	2017-01-16	Nigel	Artharg	narthargl@gizmodo.com	31 Mcbride Point	22301	Virginia	560225-6965-2870-0000	$496.00
1022	2017-01-17	Rickard	Trenholm	rtrenholmm@cbslocal.com	93 Hoepker Parkway	70593	Louisiana	3541-5241-5383-9970	$760.00
1023	2017-01-18	Juditha	Dwane	jdwanen@vk.com	7914 Eliot Lane	14276	New York	5456-4410-0914-3180	$474.00
1024	2017-01-19	Susan	Ilden	sildeno@aol.com	25204 Huxley Road	21684	Maryland	3574-8586-6367-9920	$ 83.00
1025	2017-01-20	Abbey	Triggle	atrigglep@google.com.au	47 Debra Pass	74184	Oklahoma	3538-6047-6315-7710	$513.00
1026	2017-01-21	Zsazsa	Dunster	zdunsterq@nature.com	7 Gerald Alley	40576	Kentucky	3562-0325-7709-3490	$952.00
1027	2017-01-22	Grantham	Friatt	gfriattr@seattletimes.com	774 Prairieview Circle	29225	South Carolina	3571-1171-9476-8780	$942.00
1028	2017-01-22	Ross	Gaudin	rgaudins@samsung.com	3102 Loeprich Trail	68197	Nebraska	5108-7578-4665-2710	$572.00
1029	2017-01-22	Aluino	Drover	adrovert@dagondesign.com	2717 Northridge Avenue	72199	Arkansas	670999-3171-8848-0000	$318.00
1030	2017-01-23	Shurlock	Braker	sbrakeru@huffingtonpost.com	30783 Jenna Alley	80945	Colorado	6331106-1894-9878-0000	$166.00
1031	2017-01-24	Glenda	Goodbody	ggoodbodyv@economist.com	720 Pierstorff Way	7522	New Jersey	36-0593-2719-1684	$412.00
1032	2017-01-24	Rollin	Reddie	rreddiew@tinypic.com	09 Gina Park	65810	Missouri	4665-9188-1324-1040	$383.00
1033	2017-01-26	Dorry	Jenks	djenksx@virginia.edu	1 Butterfield Road	85210	Arizona	3578-9195-0297-7730	$636.00
1034	2017-01-26	Patti	Emby	pembyy@weather.com	26 Hoard Drive	91210	California	3585-8243-7506-2470	$957.00
1035	2017-01-26	Nickie	Menauteau	nmenauteauz@ebay.com	035 Derek Junction	92110	California	3580-5488-8443-2070	$484.00



(Traditional Index)



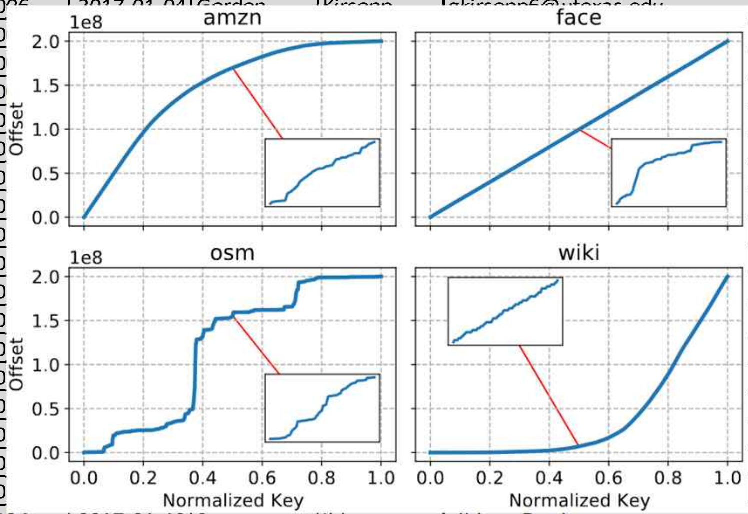
(Learned Index)

(https://dsg.csail.mit.edu/6.887/assign.php and Systems That Learn by T. Kraska)

Introduction (5/6)

- Questions
  - ✓ Always feasible?

id	date	first_name	last_name	email	address	zip	state	credit_card_nb	amount
1000	2017-01-01	Hobart	Spracklin	hspracklin0@dailymotion.com	20565 High Crossing Plaza	56372	Minnesota	4405-6975-7285-5160	$ 611.00
1001	2017-01-02	Billye	Binnion	bbinnion1@123-reg.co.uk	3698 Upham Point	20260	District of Columbia	3533-7150-7728-9850	$ 244.00
1002	2017-01-02	Johann	Brockley	jbrockley2@bizjournals.com	23844 Artisan Place	98516	Washington	67597-1193-7985-5100	$ 233.00
1003	2017-01-03	Artie	MacMenami	amacmenamin3@hao123.com	6276 Toban Trail	78759	Texas	3537-4829-6134-5000	$ 210.00
1004	2017-01-03	Delilah	O'Currigan	docurrigan4@chron.com	86016 New Castle Avenue	72199	Arkansas	3555-2017-2226-5780	$ 286.00
1005	2017-01-04	Gretta	Will	gwill5@yelp.com	0 Dottie Circle	68524	Nebraska	503844-1984-2085-5000	$ 870.00
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	64060 Scott Park	20370	District of Columbia	633332-1895-2414-5000	$ 687.00
10					94 Flor				00
10					35886				00
10					1 Bade				00
10					1 Twin				00
10					0456 N				00
10					07 Blu				00
10					7 Fairf				00
10					0995 N				00
10					80 Nov				00
10					72674				00
10					87414				00
10					2810 C				00
10					82802				00
10					89 Con				00
10					31 Mc				00
10					93 Hoe				00
10					7914 E				00
1024	2017-01-19	Susan	Ilden	sildeno@aol.com	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1					47 Debra				3.00
1					7 Gerald				2.00
1					774 Prairi				2.00
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1029	2017-01-22	Aluino	Drover	adrovert@dagondesign.com	2717 Northridge Avenue	72199	Arkansas	670999-3171-8848-0000	$ 318.00
1030	2017-01-23	Shurlock	Braker	sbrakeru@huffingtonpost.com	30783 Jenna Alley	80945	Colorado	6331106-1894-9878-0000	$ 166.00
1031	2017-01-24	Glenda	Goodbody	ggoodbodyv@economist.com	720 Pierstorff Way	7522	New Jersey	36-0593-2719-1684	$ 412.00
1032	2017-01-24	Rollin	Reddie	rreddiew@tinypic.com	09 Gina Park	65810	Missouri	4665-9188-1324-1040	$ 383.00
1033	2017-01-26	Dorry	Jenks	djenksx@virginia.edu	1 Butterfield Road	85210	Arizona	3578-9195-0297-7730	$ 636.00
1034	2017-01-26	Patti	Emby	pembyy@weather.com	26 Hoard Drive	91210	California	3585-8243-7506-2470	$ 957.00
1035	2017-01-26	Nickie	Menauteau	nmenauteauz@ebay.com	035 Derek Junction	92110	California	3580-5488-8443-2070	$ 484.00



Various real-world integer/decimal dataset (Benchmarking learned indexes, VLDB`20)

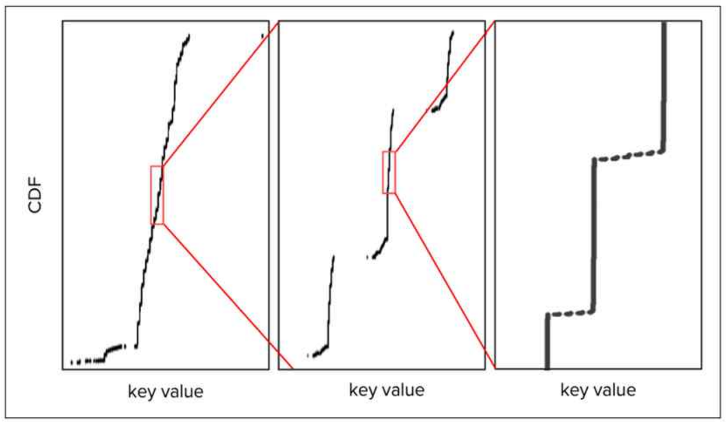
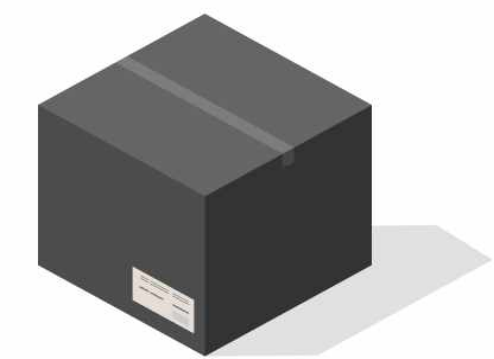


Figure 1-2: Repeated zooming on the CDF of the Wiki dataset reveals its fractally stepwise nature.

Real world string dataset (wiki) (Efficient Learned String Indexing, AIDB`21)

Introduction (6/6)

- Key insight of Learned Index



Traditional data structures (typically) make no assumptions about the data.

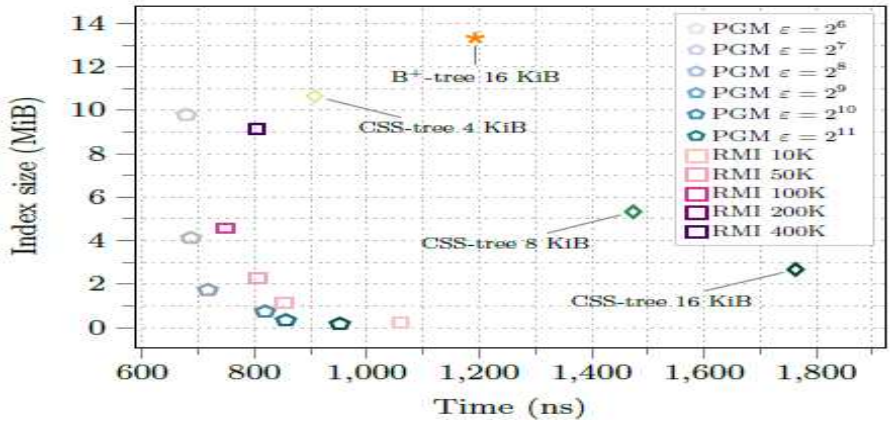
But knowing the data distribution might allow for significant performance gains and might even change the complexity of data structures.

O(log n) → O(1) for lookups
O(n) → O(1) for memory usage

(Source: AWS Data Insights Day: How will AI revolutionize databases?)

Motivation (1/4)

- Issues
  - ✓ How to learn: RMI, PGM-index, RS, …
  - ✓ Updatable: Tree-merge, Sparse-node, Delta-merge, …
  - ✓ Multi-dimensional: Linearization, Hierarchical, Distance-based, …



(PGM-Index, VLDB'20)

Index	Parameters
ALEX	Max inner/data node size: 16MB Min/avg/max node density: 0.6/0.7/0.8
ALEX+	Max inner/data node size: 16MB/512KB Min/avg/max node density: 0.6/0.7/0.8
LIPP(+)	Node density: 0.5; max node size: 16MB Subtree inserted/conflict ratio: 2/0.1
PGM-Index[2]	Error bound: 16
XIndex	Error bound: 32; delta size: 256 Error tolerance: 1/4; max number of models per group: 4
FINEdex	Error bound: 32

(Updatable Learned Index Ready, VLDB'22)

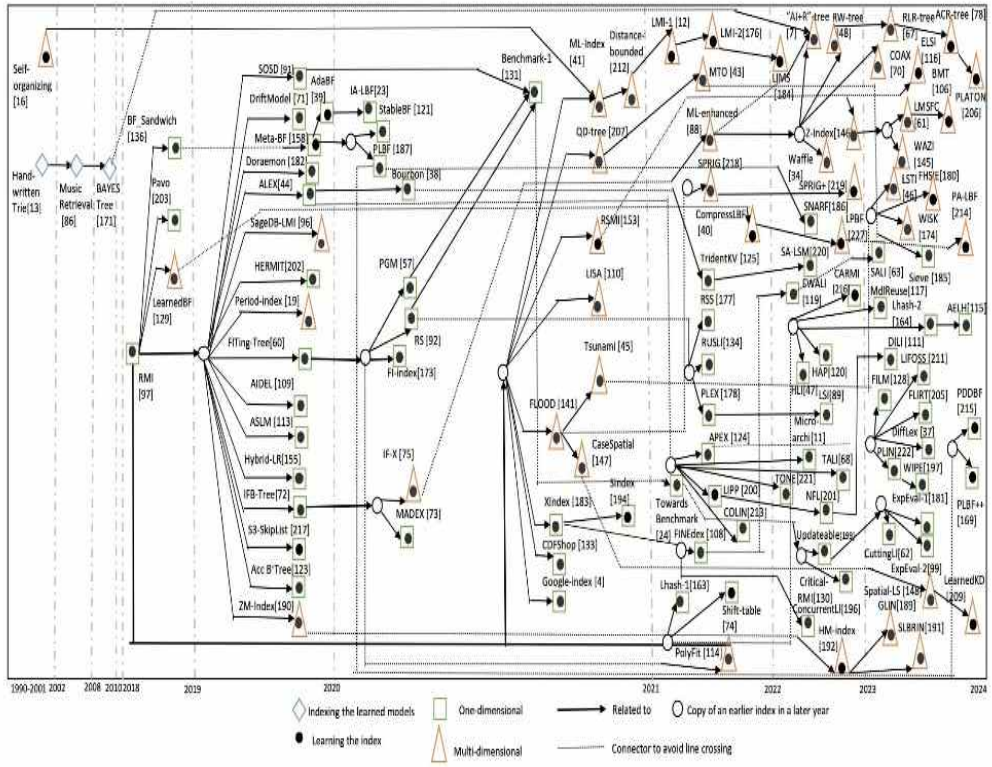
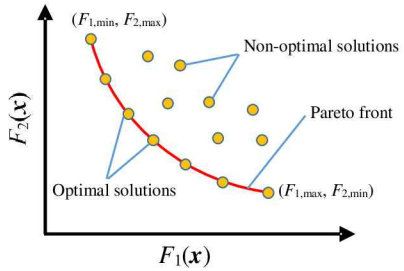


Fig. 4. Timeline of the evolution of learned indexes. Lines connecting between the various learned multi-dimensional indexes reflect dependence of these indexes on earlier work.

(Learned Indexes for Muti-dimensional,arXiv'24)

Motivation (2/4)

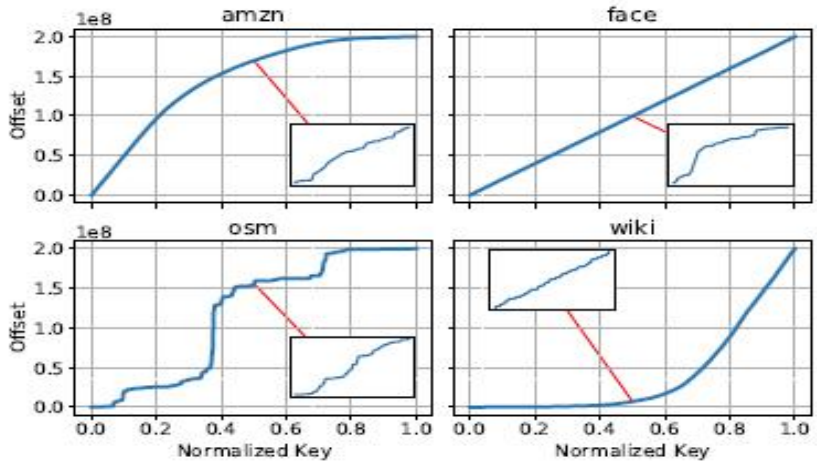
- From “Learned Index Benchmark” (VLDB’20)
  - ✓ Propose a benchmark for fair comparison
    - Between learned and traditional indexes
      - Learned index: RMI, PGM, RS (indexes for immutable)
      - Traditional index: Btree, IBtree, FAST, ART, …
    - 4 Datasets
  - ✓ Metric: Pareto optimal
    - A state at which resources in a given system are optimized in a way that one dimension cannot improve without a second worsening



(Pareto Optimal)

Method	Updates	Ordered	Type
PGM [12]	Yes	Yes	Learned
RS [17]	No	Yes	Learned
RMI [18]	No	Yes	Learned
BTree [6]	Yes	Yes	Tree
IBTree [14]	Yes	Yes	Tree
FAST [15]	No	Yes	Tree
ART [19]	Yes	Yes	Trie
FST [32]	Yes	Yes	Trie
Wormhole [30]	Yes	Yes	Hybrid hash/trie
CuckooMap [5]	Yes	No	Hash
RobinHash [2]	Yes	No	Hash
RBS	No	Yes	Lookup table
BS	No	Yes	Binary search

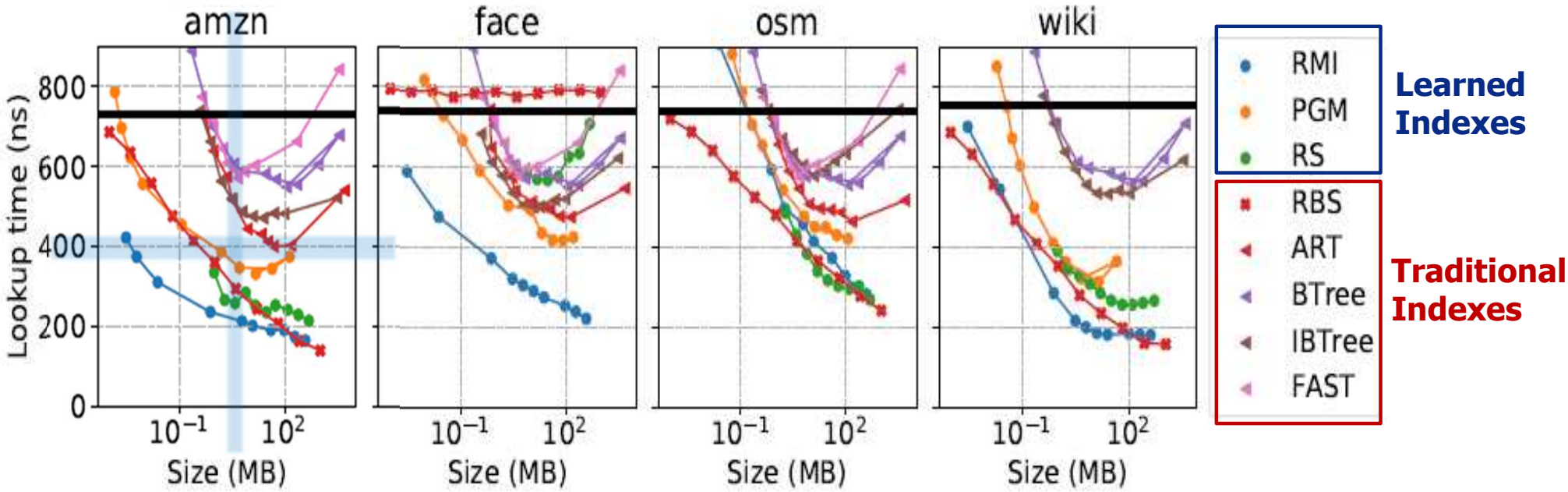
Table 1: Search techniques evaluated



(Benchmarking Learned Index, VLDB’20)

Motivation (3/4)

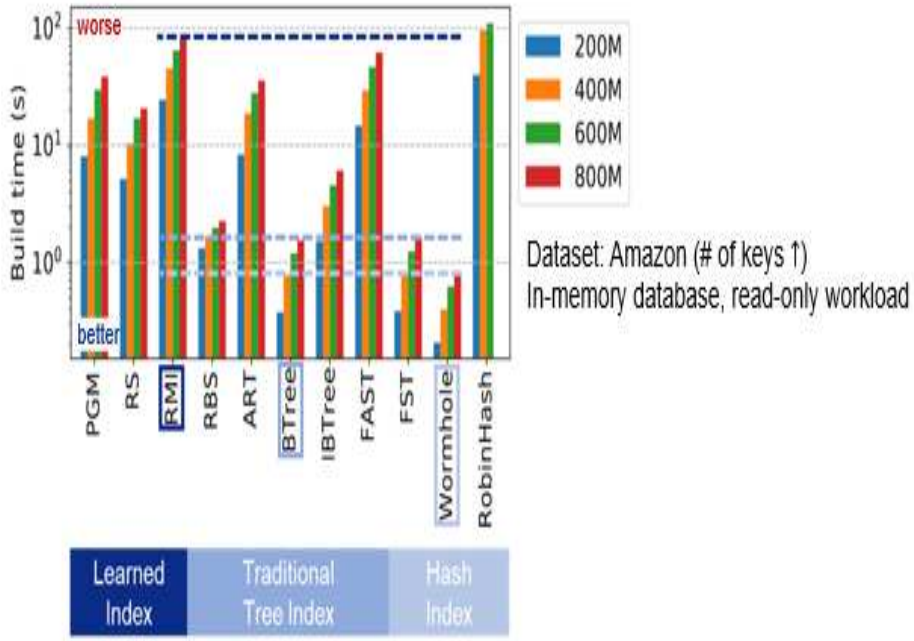
- Observation1: size-lookup analysis
  - ✓ Learned Indexes are pareto optimal for size-lookup viewpoint
    - No alternative exists that has both a smaller size and lower latency
  - ✓ Example: amazon workload case
    - When latency is 400ns, learned indexes are up to 4,000x smaller in size
    - When index size is 1MB, learned indexes are up to 3x faster in latency



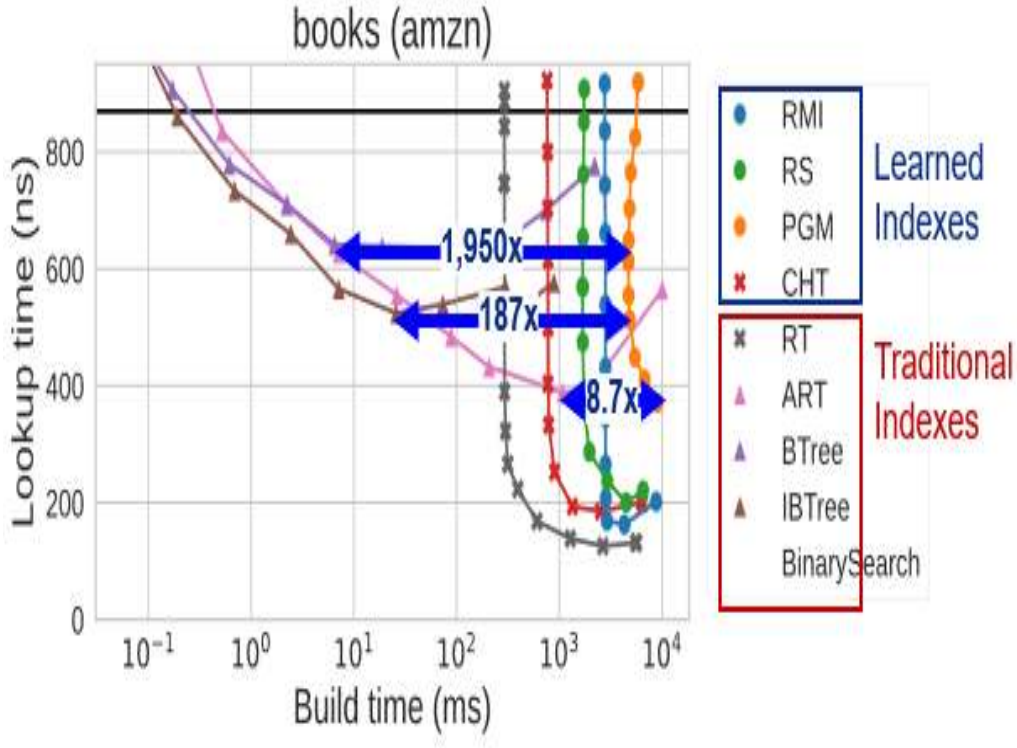
(Benchmarking Learned Index, VLDB'20)

Motivation (4/4)

- Observation2: Build time (learning time) analysis
  - ✓ Reveal learned index limitation: Longer build time
    - Benchmarking paper: 10x slower in build-time
    - Our own analysis: not pareto optimal for build-lookup viewpoint
      - Trend: can not reduce build time even less QoS (lookup time) requirement



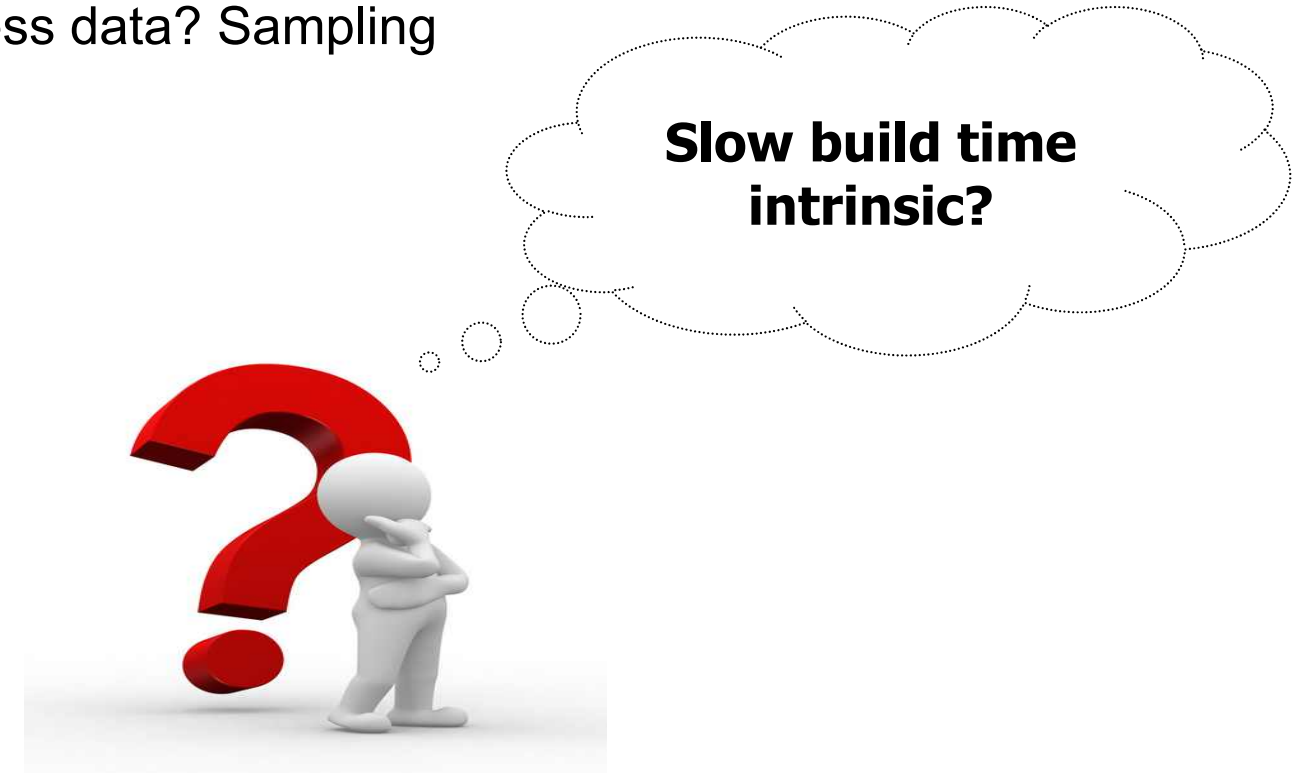
(Benchmarking Learned Index, VLDB'20)



(Our experiments)

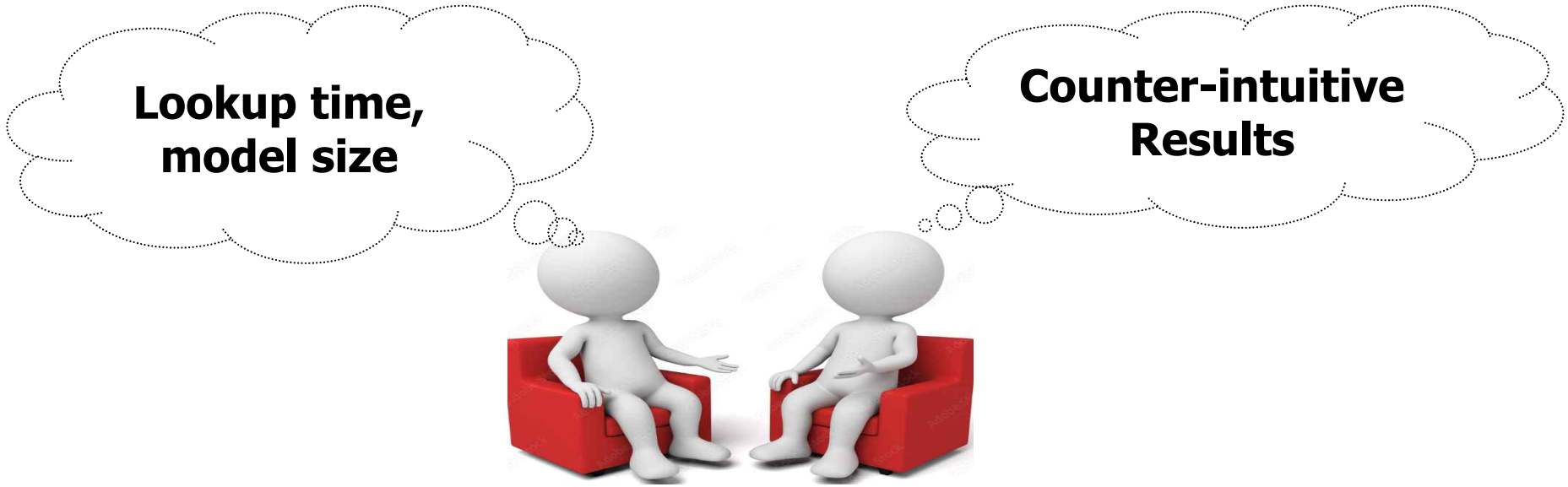
Challenges (1/3)

- Question
  - ✓ Is the slow build time intrinsic in Learned Indexes (unavoidable)?
  - ✓ Existing build time improvement approaches
    - Lightweight learning: cubic → linear, 2-layer recommendation in RMI, quantization, …
    - Delayed learning: Gapped array, Delta-merging, …
    - How about less data? Sampling



Challenges (2/3)

- Question
  - ✓ How about less data? Sampling
    - It may hurt learning accuracy, eventually degrade lookup time
      - Can provide comparable lookup time via the error-bound property ➔ sampling and error bound interaction
    - It may decrease the model size
      - Depends on learned indexes (same or rather increasing in some cases) ➔ complex trade-offs
    - How to compare with the traditional indexes?

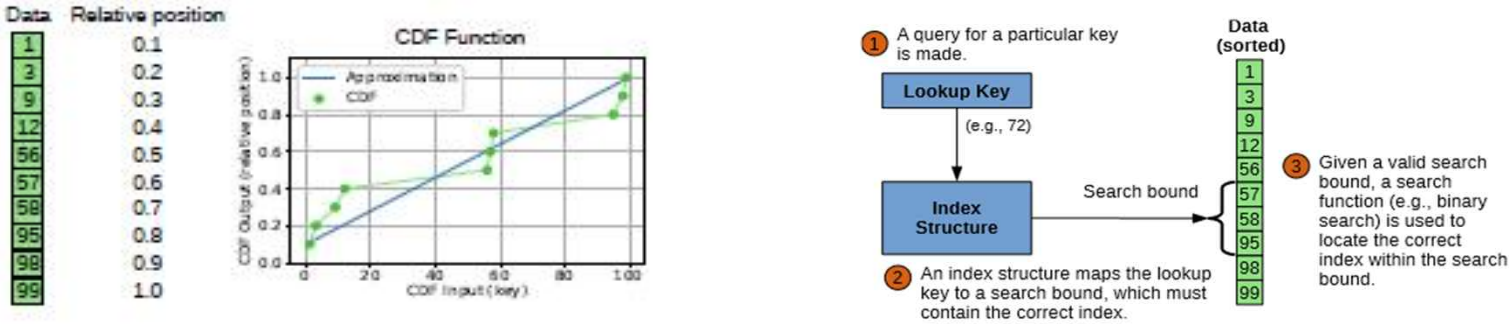


Challenges (3/3)

- Basic Idea
  - ✓ Apply sampling to reduce the building overhead
    - Sampling may seem simple and even naïve ➔ But, complex indeed!
- Three Challenges
  - ✓ 1. Preserving error-bound property
    - Make use of systematic sampling
    - Propose a new Sample EB-PLA (Error-Bounded Piecewise Linear Approximation)
  - ✓ 2. Complex trade-offs
    - Applying sampling to representative learned indexes
    - Analyze Build time/Model size/Lookup time with the consideration of Micro-architecture events (cache, branch prediction, # of instructions)
  - ✓ 3. Fair comparison
    - Unfair comparison in existing open-sources (e.g. with/without data copy before learning) ➔ anomaly results ➔ hurdle to identify real problems
    - New benchmark: Unified sampling methods and implementations

Solutions (1/8)

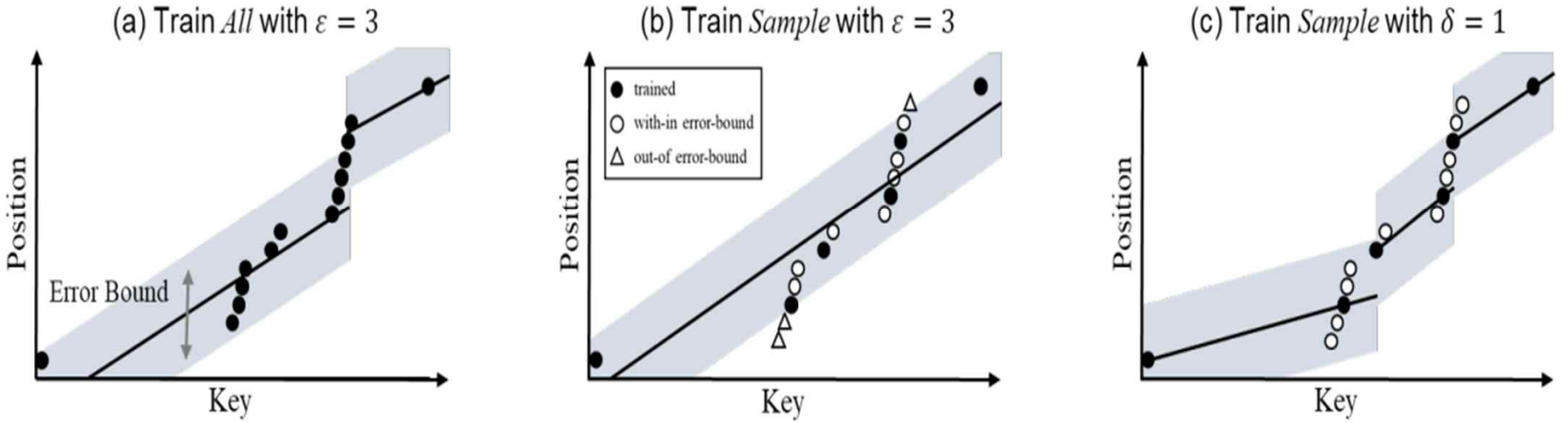
- 1. Preserving error-bound property
  - ✓ What is Error-bound?
    - Basic
      - Data: A sorted array D of integer keys
      - Index: maps a lookup key x to a bound that contains the “lower bound” of x
    - Building (training, learning)
      - Estimate data distribution ➔ CDF (Cumulative Distribution Function)
      - Make use of diverse ML techniques (e.g. Linear Regression)
    - Lookup: two steps
      - Prediction (inference): estimates the position of x in D as p
      - Correction (last-mile search): based on p, nearby keys are searched to find the exact position of x
    - Error-bounded property
      - $\forall k \in D$, $Error(k) \leq \varepsilon$, k exist in correction range (= $[p - \varepsilon, p + \varepsilon]$)



(Benchmarking Learned Index, VLDB’20)

Solutions (2/8)

- 1. Preserving error-bound property
  - ✓ How to achieve error-bound in existing learned indexes?
    - Basic idea: error-bound aware learning
      - If not guarantee a given error, employ the divide-and-conquer approach
    - Example: EB-PLA (Error-Bounded Piecewise Linear Approximation)
      - While training, ensure all errors are less than ε.
      - Otherwise, form a new segment dynamically ➔ Figure (a)
  - ✓ Effect of Sampling
    - Cannot guarantee an error-bound of un-trained data ➔ Figure (b)



Solutions (3/8)

- 1. Preserving error-bound property
  - ✓ Our approach: Sample EB-PLA
    - Support error bound with Sampling
    - $For\ \forall k \in D,\ Error(k) \leq \varepsilon \rightarrow For\ \forall s \in S,\ Error(s) \leq \delta\ (=\varepsilon - I + 1)$
      - 1) Systematic sampling: equal-distance with distance I
      - 2) Train sampled data with the smaller error bound δ ($\delta = \varepsilon - I + 1$)
      - 3) Trained model can guarantee error bound ε for all data (even unsampled)
    - Proof: refer to our paper

THEOREM 3.1. *Assume an algorithm constructs a EB-PLA model M with all s in S and $Pos(s)$ in D with the parameter $\delta(\geq 0)$. M is a monotonically increasing function that ensures $\forall s \in S, Error(s) \leq \delta$ in D. Then, M guarantees that $\forall k \in D, Error(x) \leq \varepsilon(= \delta + I - 1)$ in D.*

PROOF. Assume k not in S, which lies between two consecutive sampled keys, s_{lo} and s_{hi}. (i.e., $s_{lo} < k < s_{hi}$, and $hi = lo + 1$.) $Pos(k)$ is at most $I - 1$ different from both $Pos(s_{lo})$ and $Pos(k_{hi})$. From the above relations, we can bound $Pos(k)$ in terms of $Pos(s_{lo})$, $Pos(s_{hi})$ as following:

$$Pos(s_{hi}) - I + 1 \leq Pos(k) \leq Pos(s_{lo}) + I - 1.$$

Since M is a monotonically increasing function, $Pred(s_{lo}) \leq Pred(k) \leq Pred(s_{hi})$. Given that M ensures $Error(s) \leq \delta$, we can bound $Pred(k)$ in terms of $Pos(s_{lo})$, $Pos(s_{hi})$ and δ.

$$Pos(s_{lo}) - \delta \leq Pred(k) \leq Pos(s_{hi}) + \delta$$

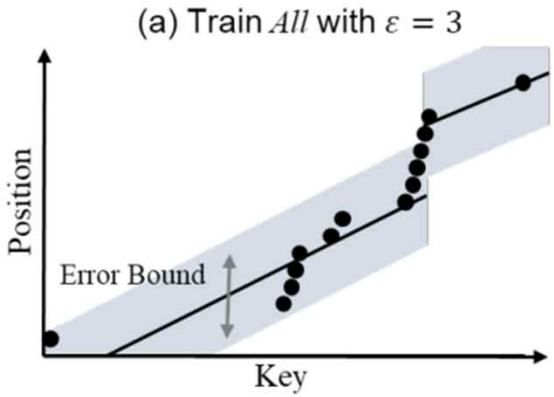
Through this, the prior bound of $Pred(k)$ in relation to $Pos(s_{lo})$, $Pos(s_{hi})$ and δ can be transformed into a relation to $Pos(k)$, δ and I. From the above relations, we get the following inequality, which proves the theorem:

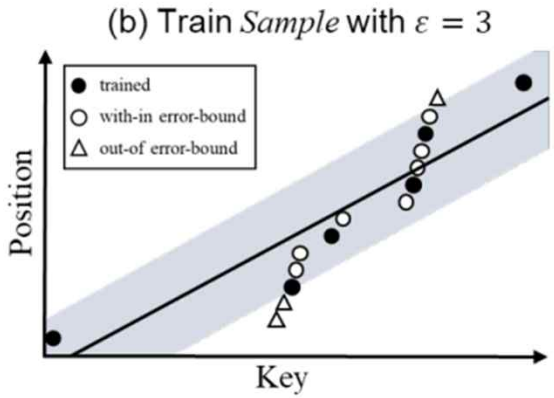
$$Pred(k) - \delta - I + 1 \leq Pos(k) \leq Pred(k) + \delta + I - 1.$$

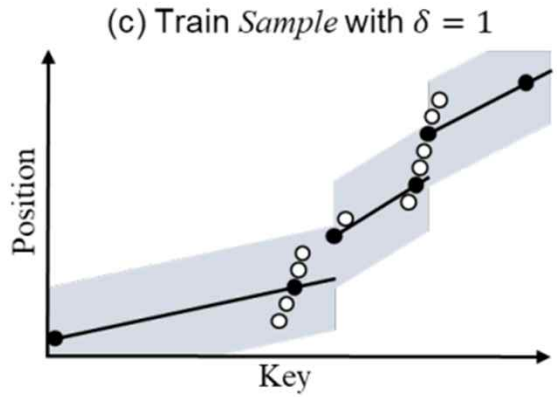
Therefore, $Error(k)$ is guaranteed to be no greater than ε, and so is $Error(s)$. Finally, we get

$$Error(k) = |Pos(k) - Pred(k)| \leq \delta + I - 1 = \varepsilon$$

□







Solutions (4/8)

- 2. Applying sampling to representative learned indexes
  - ✓ Four cases
    - RMI (Recursive Model Index): SIGMOD'18
    - PGM-Index (Piecewise Geometric Model index): VLDB'20
    - RS (RadixSpline): AIDM'20
    - Compact Hist-Tree: CIDR'21

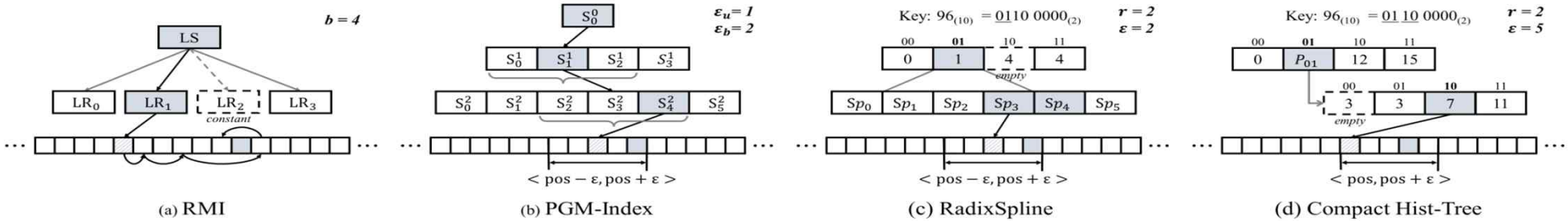


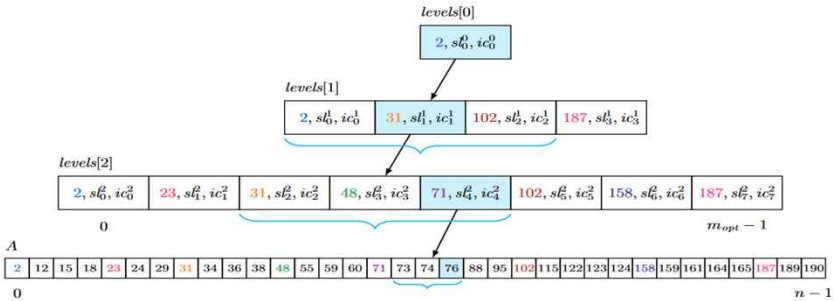
Figure 1: Structure of indexes: RMI, PGM, RS, and CHT.

Index	Model				Building			Correction		
	Top (Upper) Layer	Bottom Layer	# of Layers	Param.	Training Algorithm	Segmentation	Order	Error Bounded	Search Algorithm	Correction Range
RMI [13]	Linear Spline	PLR	2	b	Linear Least Squares	fixed	↓	X	Exponential	-
PGM [8]	EB-PLR	EB-PLR	≥ 2	ε_u , ε_b	Optimal PLR [39]	dynamic	↑	O	Binary	[p-ε,p+ε]
RS [11]	Histogram	EB-PLS	2	r, ε	Greedy Spline Corridor [29]	dynamic	↑	O	Binary	[p-ε,p+ε]
CHT [3]	Histogram	EB-Histogram	≥ 1	r, ε	EB-Histogram [3]	fixed & dynamic	↓	O	Binary	[p,p+ε]

Table 1: Summary of the Indexes.

Solutions (5/8)

- 2. Applying sampling to PGM-Index
  - ✓ Structure: A multi-layered learned index
    - Each layer: Error-bounded piecewise linear regression (EB-PLR)
      - Guarantees a predefined error bound (ε) in all predictions
  - ✓ Building: bottom-up ➔ sampled keys
    - Bottom Layer: EB-PLR ➔ Sample EB-PLR
      - If an additional data point is out of the given error bound → completes the current segment → creates an additional line segment on demand
    - Upper Layer: EB-PLR
      - Repeats until it reaches the top layer, which has a single segment.
  - ✓ Lookup
    - Prediction
      - Top layer ➔ Bottom layer: the position of the key in the dataset is predicted.
    - Correction
      - Binary search the key only within the correction range



(Structure)

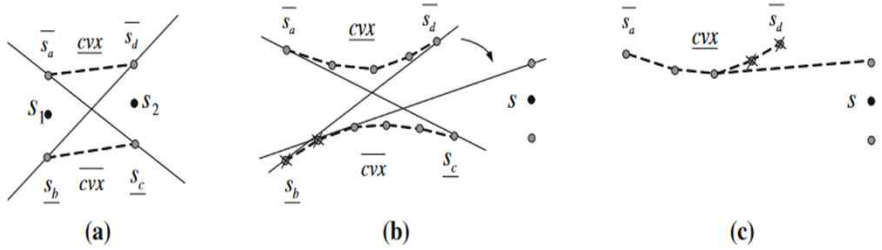


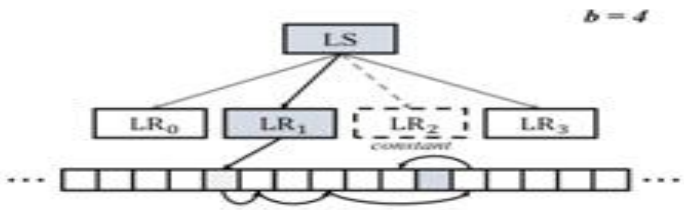
Fig. 7 Optimal algorithm: a Initialization; b Updating of extreme slopes; c Updating of convex hull

(Building)

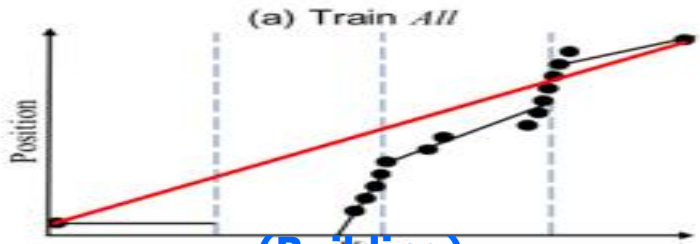
(PGM-index, VLDB'20 and Maximum Error-bounded Piecewise Linear Representation, VLDB'14)

Solutions (6/8)

- 2. Applying sampling to RMI
  - ✓ Structure: A multi-layered learned index supporting various sub models
    - Two-level RMI: known as best suited when all keys fit in memory
  - ✓ Building: top-down ➔ sampled keys
    - Top Layer: Linear Spline ➔ branch factor (unchanged in this study)
      - Linear spline using the first and the last keys ➔ Divides the key range into b number of fixed ranges (fixed segmentation, b (branching factor) : # of segs.)
    - Bottom Layer: Linear Regression
      - Each segment is trained to minimize the mean squared error (MSE) of data
      - No Error Bound
  - ✓ Lookup
    - Prediction
      - Top: Choose a segment using b ➔ Bottom: Estimate a position
    - Correction
      - Exponential search based on the predicted position (no error bound)



(Structure)

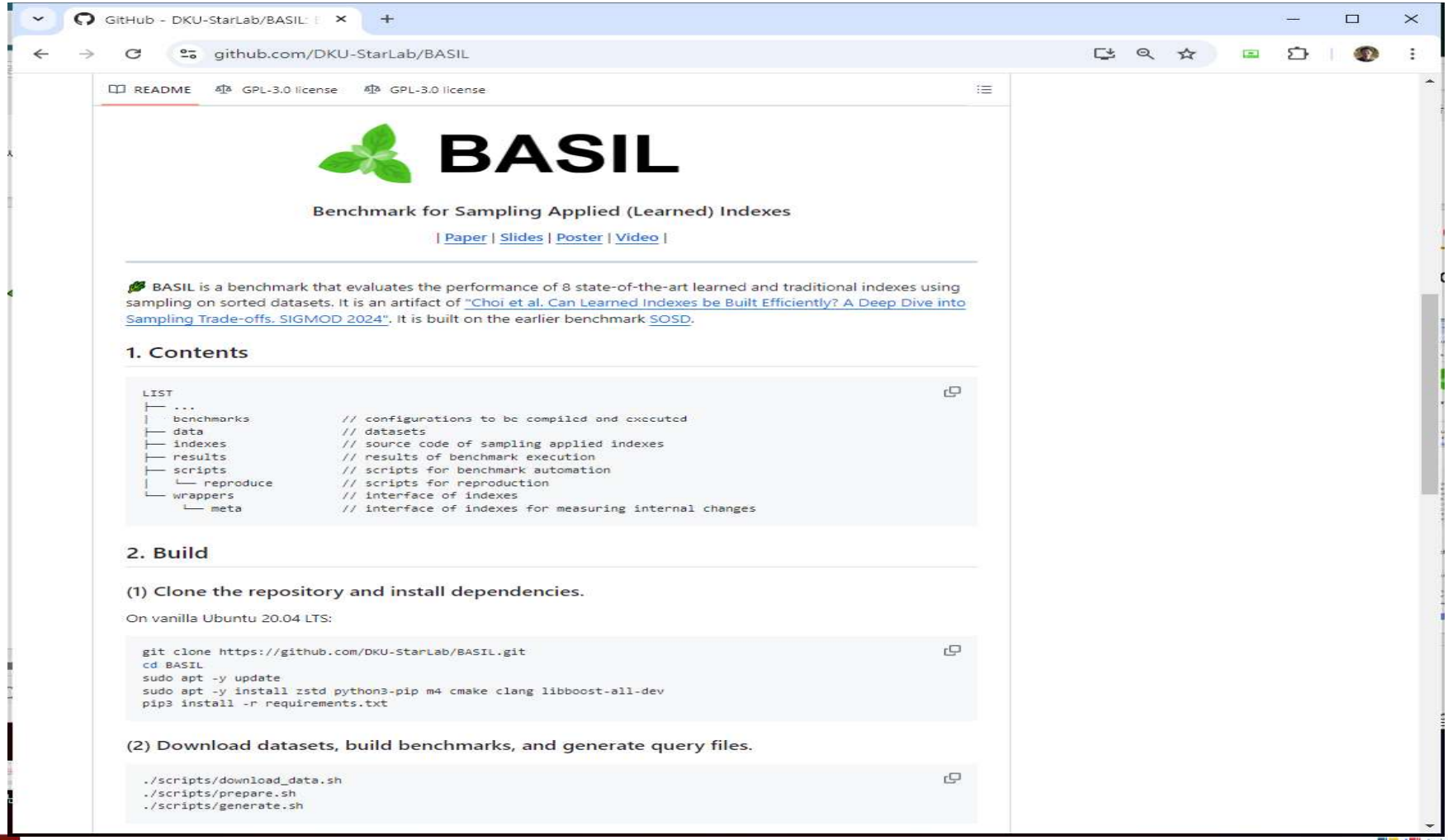


(Building)

(The Learned Index, SIGMOD'18)

Solutions (7/8)

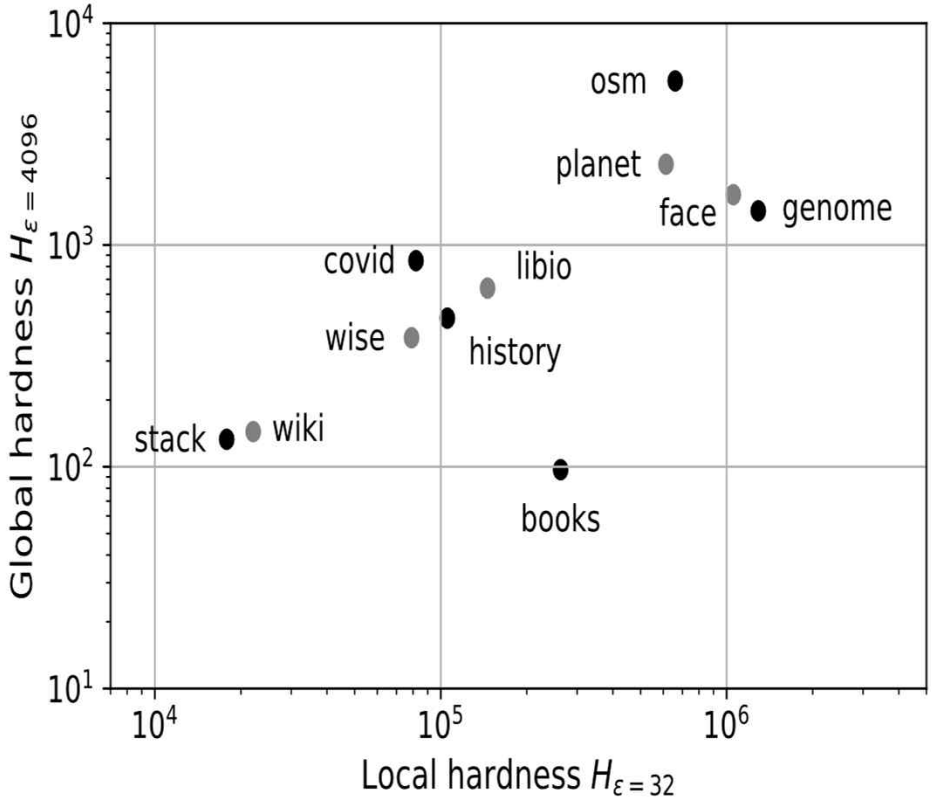
- 3. New open-sourced benchmark for Fair comparison
  - ✓ BASIL: Benchmark for Sampling Applied Learned Indexes
    - https://github.com/DKU-StarLab/BASIL



Solutions (8/8)

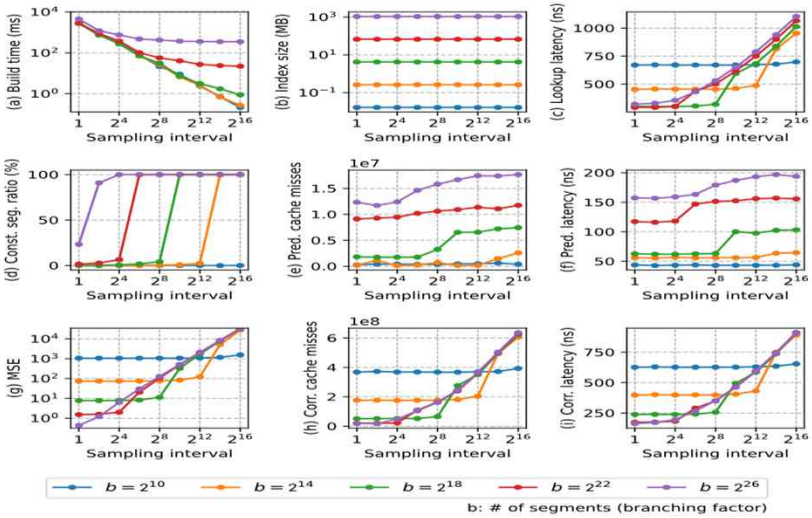
- 3. New open-sourced benchmark for Fair comparison
  - ✓ Sampled Indexes
    - 3 Learned indexes + 2 Histogram-based + 3 Traditional indexes
  - ✓ Datasets
    - 10 classified using local hardness and global hardness

Type	Index
Learned	sRMI
Learned	sPGM / sRS
Histogram	sCHT
Histogram	sRT
Tree-based	sART / sB+-Tree/ sIB-Tree

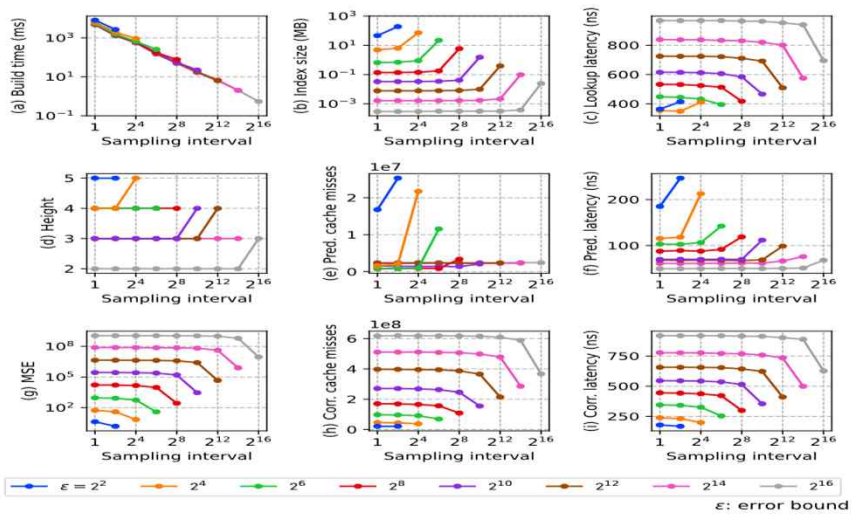


Evaluations (1/5)

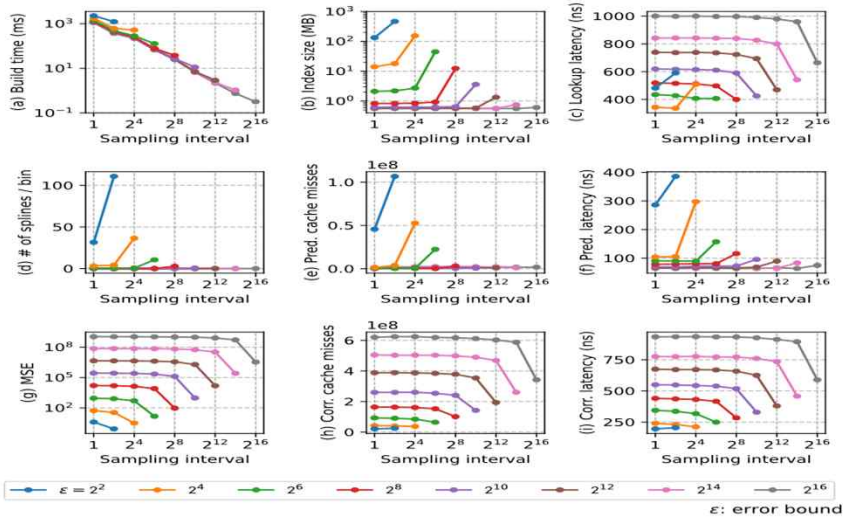
- 1. Trade-offs analysis: all



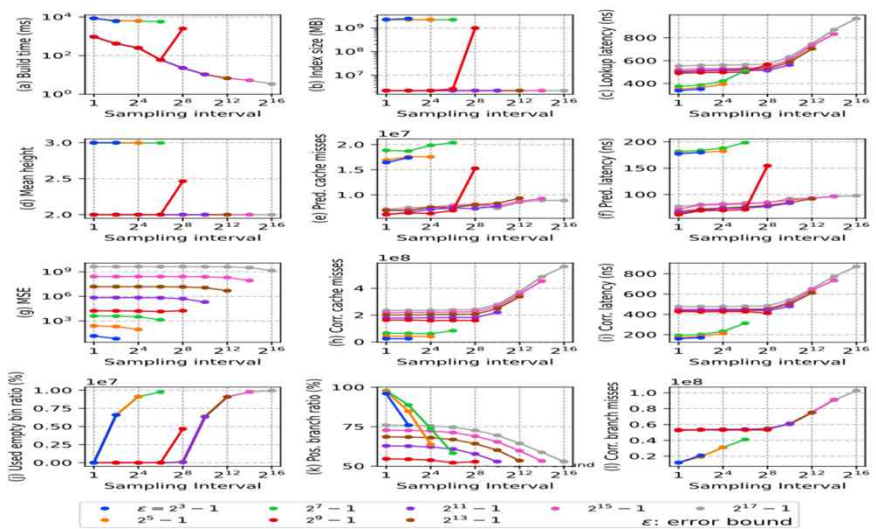
(sRMI)



(sPGM)



(sRS)



(sCHT)

Evaluations (2/5)

- Trade-offs analysis: sPGM
  - ✓ Build time: Figure 7(a)
    - Improve as Sampling Interval increase
    - Similar to all error bound
  - ✓ Index size: Figure 7(b)
    - Increase as Interval increase
      - Due to smaller error bound
      - Extreme case: (error bound == Interval) ➔ Increase height (Figure 7 (d))
  - ✓ Lookup latency: Figure 7(c)
    - Stable (or slightly enhanced) as Interval increase
    - Prediction (Figure 7(f))
      - Similar to the index size
      - Extreme case: increase due to huge cache miss (Figure 7(e))
    - Correction latency (Figure 7(i))
      - Enhance due to reduced MSE (Figure 7(g)) and cache miss (Figure 7(h))

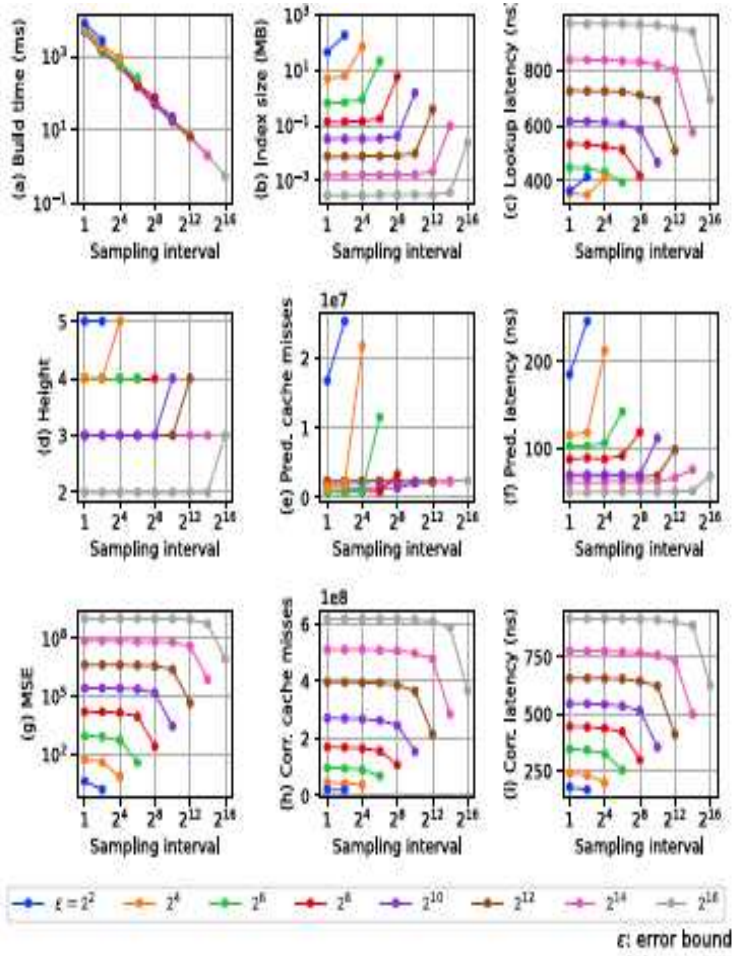


Figure 7: Performance impact of sampling on sPGM on *history* dataset.

- ✓ As the sampling interval increases, the build time decreases (from 8.2 s to 543 µs).
- ✓ Index size and lookup latency: 1) comparable to non-sampling case (good feature), 2) aggressive sampling do harmful for index size.

Evaluations (3/5)

- Trade-offs analysis: sRMI
  - ✓ Build time: Figure 6(a)
    - Improve as Sampling Interval increase
    - Early saturated with larger branching factor
  - ✓ Index size: Figure 6(b)
    - Same regardless Sampling Interval
      - Due to same branch factor
      - Increase constant segments (0 or 1 key) with larger b (Figure 6(d))
  - ✓ Lookup latency: Figure 6(c)
    - Stable when const. segs. are small
      - Degrade when const. segs. increase
    - Prediction latency (Figure 6(f))
      - Depend on the # of const. segs, due to cache miss (Figure 6(e)
    - Correction latency (Figure 6(i))
      - Depend on the # of const. segs, due to MSE (Figure 6(g)) and cache miss (Figure 6(h))

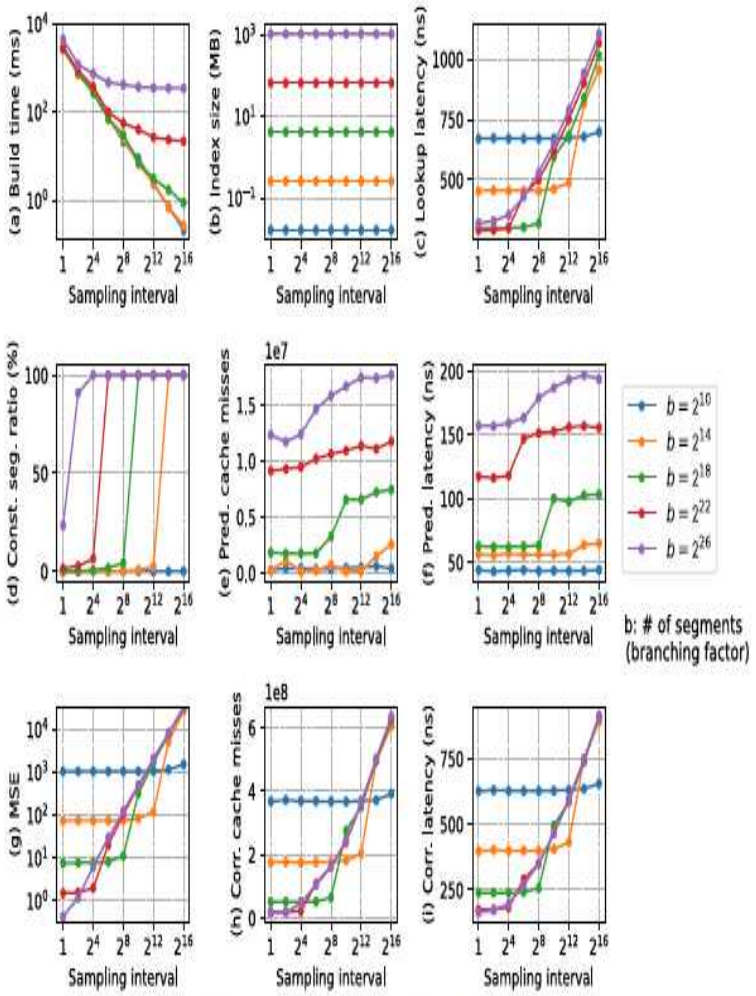
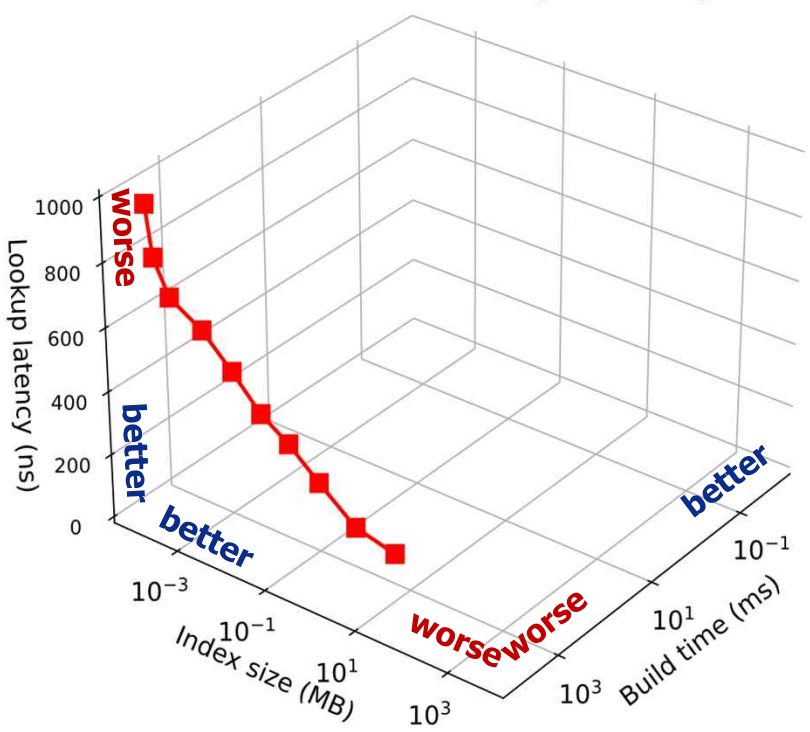


Fig. 6. Performance impact of sampling on sRMI on *history* dataset.

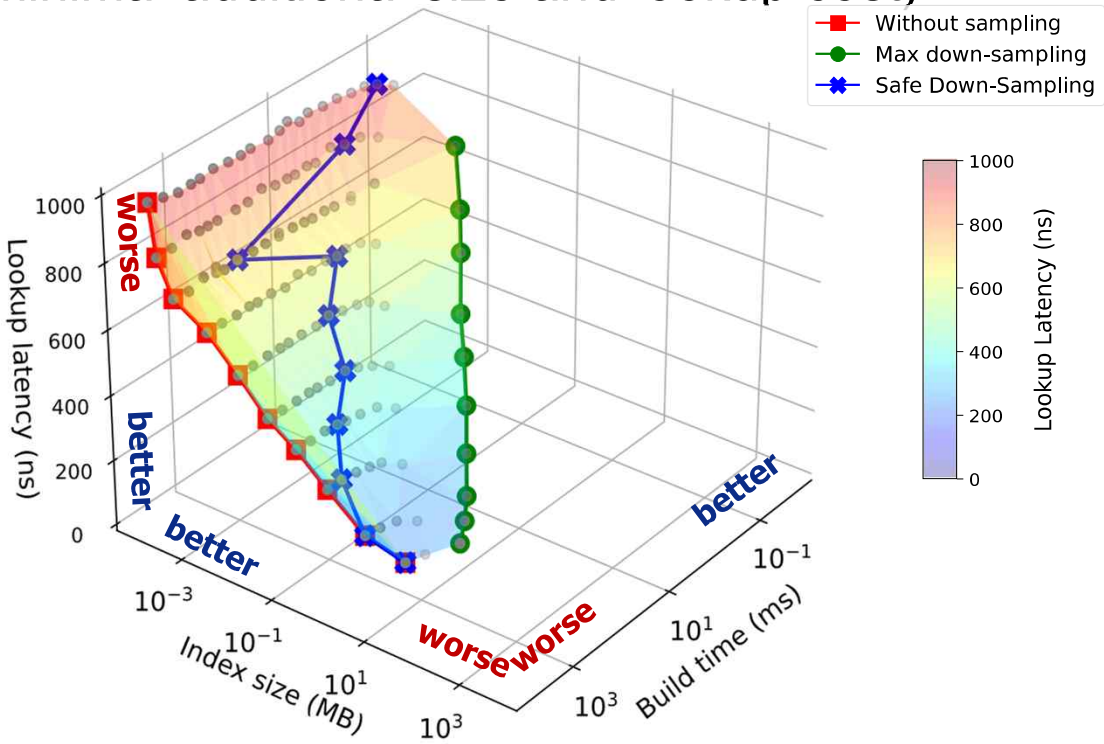
- ✓ As the sampling interval increases, the build time decreases.
- ✓ Index size: 1) same, 2) depend on model (same for RMI, increased for PGM)
- ✓ Lookup latency: 1) stable when the const. segs. are small (good feature), 2) aggressive sampling harmful (or Sampling needs to rethink training algorithms to reduce const. segs.)

Evaluations (4/5)

- 2. Broaden design space
  - ✓ Sampling introduces new design space: size vs. lookup ➔ size vs. lookup vs. build
  - ✓ Provide safe sampling intervals
    - Configuration where it can reduce build time without increasing size and lookup cost (or with minimal additional size and lookup cost)



(Without EB-Sampling)



(With EB-Sampling)

Evaluations (5/5)

- 3. Pareto optimal
  - ✓ Learned indexes are pareto optimal for build-lookup viewpoint, too
    - Both average and 99.9 tail latency
    - For example: history workload
      - When the lookup latency is 400ns, learned indexes are by up to 320x faster in build

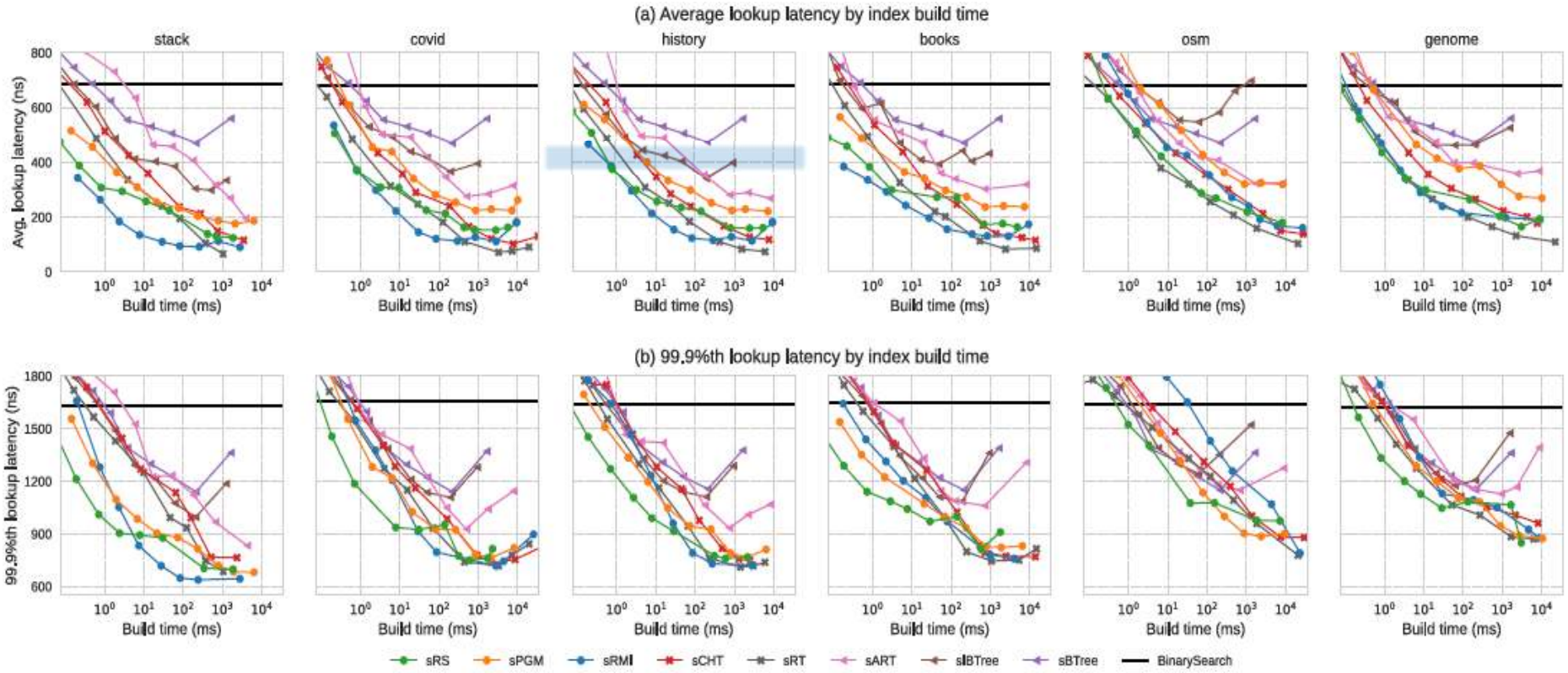
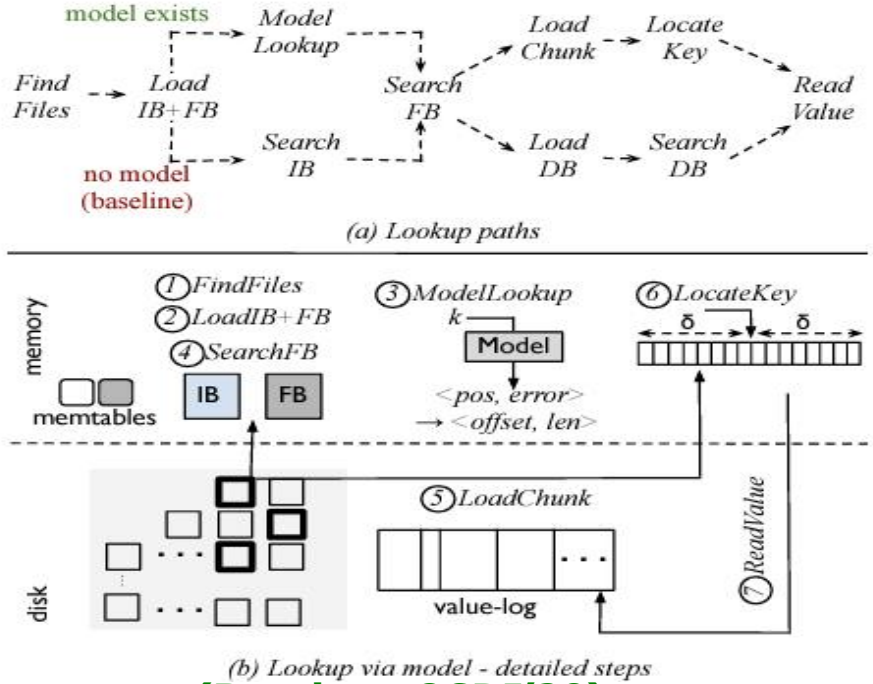


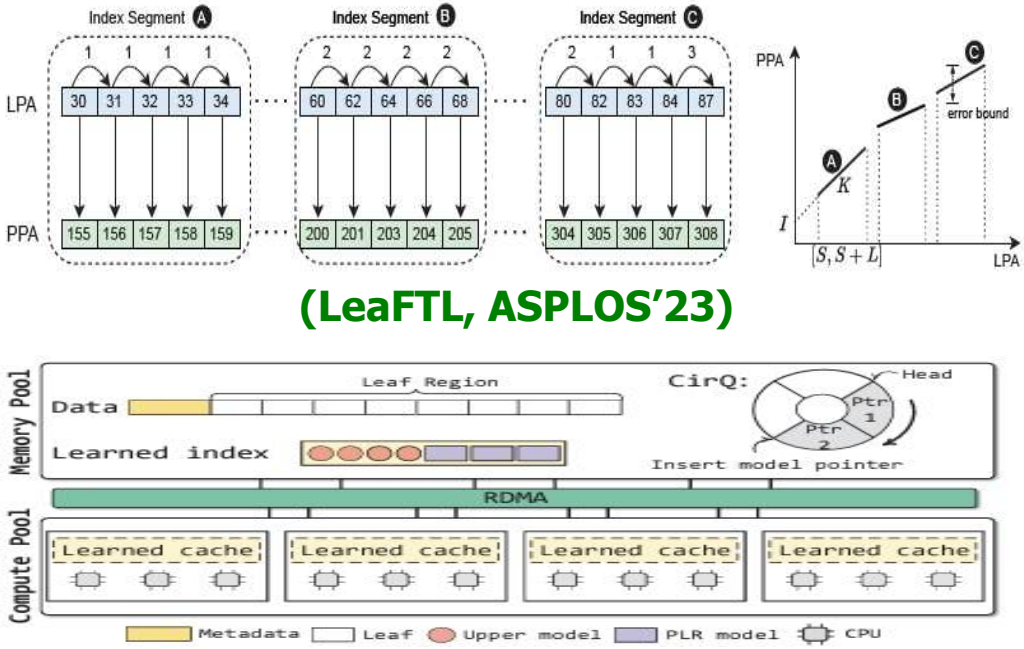
Figure 12: Average latency and 99.9 percentile tail latency by the index build time.

Conclusion

- Learned index
  - ✓ Employing machine learning for indexing
  - ✓ Promising for memory usage and lookup time
  - ✓ Our study reveals that it can be built-efficient using sampling
- Applying learned index into real applications
  - ✓ 1) Key-Value Store, 2) FTL, 3) RDMA, 4) Disaggregated system, …



(Bourbon, OSDI'20)



(LeaFTL, ASPLOS'23)

(ROLEX, FAST'23)

Discussion