A Learned Index Journey

Minguk Choi
System Software Laboratory, Master's Student

Before study the learned index

- Studying ***Index*** in a System Software Laboratory might not be enjoyable.

- When I was an undergraduate researcher …
  - My research area was ***Key-Value Store*** .
  - Professor told me to join ***the Kernel Study***, which I didn’t want to.
  - ***The kernel memory system*** I had studied earlier proved to be a great help while writing learned index paper.

- Even if it's not particularly appealing
  - I hope you to **study the (learned) index hard!**

Before study the learned index

- Minguk Choi

- Education
  - MS, Department of AI-based Convergence, Mar 2023 - Aug 2024
  - BS, Department of Software Science, Mar 2017 - Feb 2023,
    - **Summa Cum Laude (Salutatorian)**

- Publications
  - **Minguk Choi**, Seehwan Yoo and Jongmoo Choi. “Can Learned Index be Build-Efficient? A Deep Dive into Sampling Trade-Offs.” **SIGMOD 2024**
  - Ramadhan, Agung Rahmat, **Minguk Choi**, Yoojin Chung, and Jongmoo Choi. "An Empirical Study of Segmented Linear Regression Search in LevelDB." Electronics 12, no. 4 (2023): 1018.

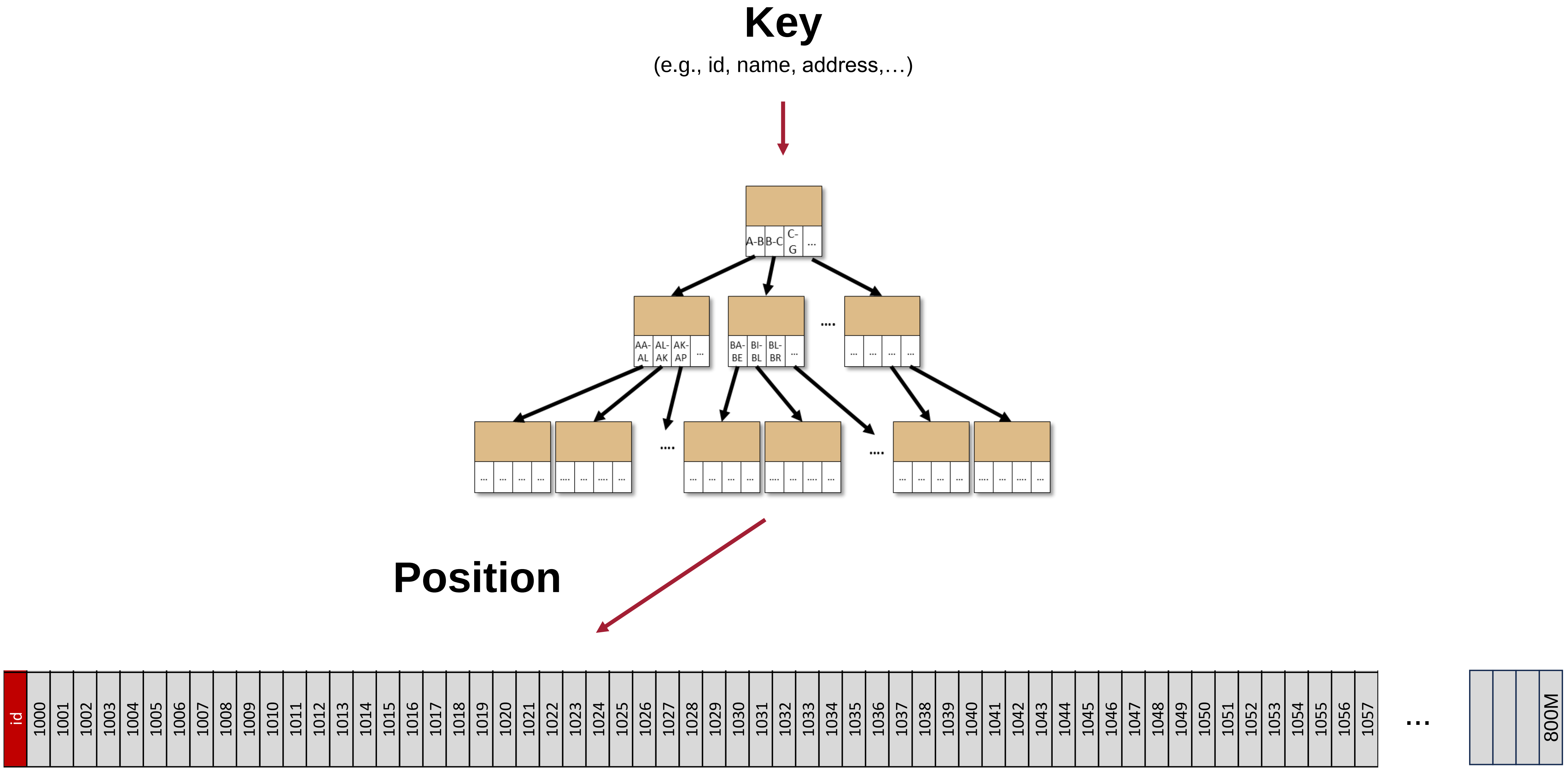
- Teaching Experience
  - System Programming, Sep 2023 - Dec 2023
  - Operating System practice, Sep 2023
  - **LevelDB Study, Jul 2022 - Aug 2022**

1. What is learned index?

Exploiting the 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	nartharg l@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

Traditional B-Tree Index



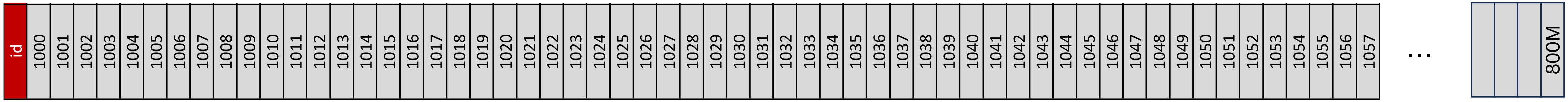
Traditional B-Tree Index

Key

(e.g., id, name, address,…)

array[id – 1000]

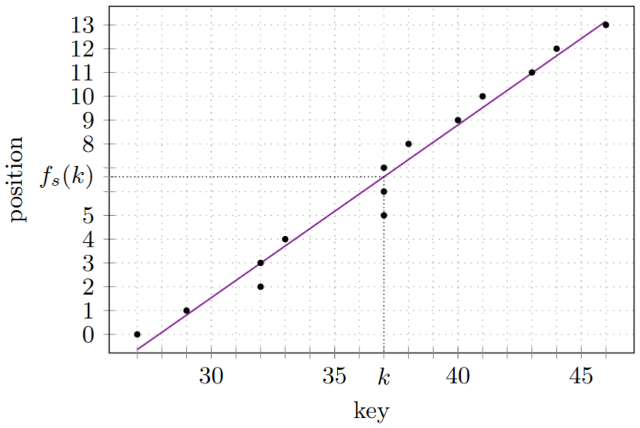
Position



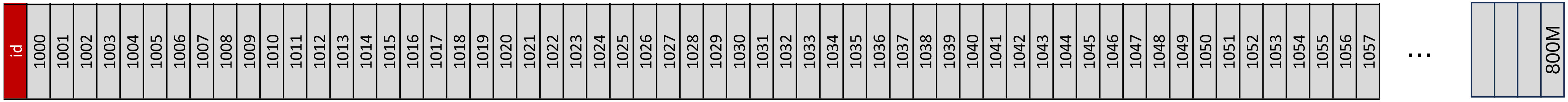
Learned Index

Key

(e.g., id, name, address,…)

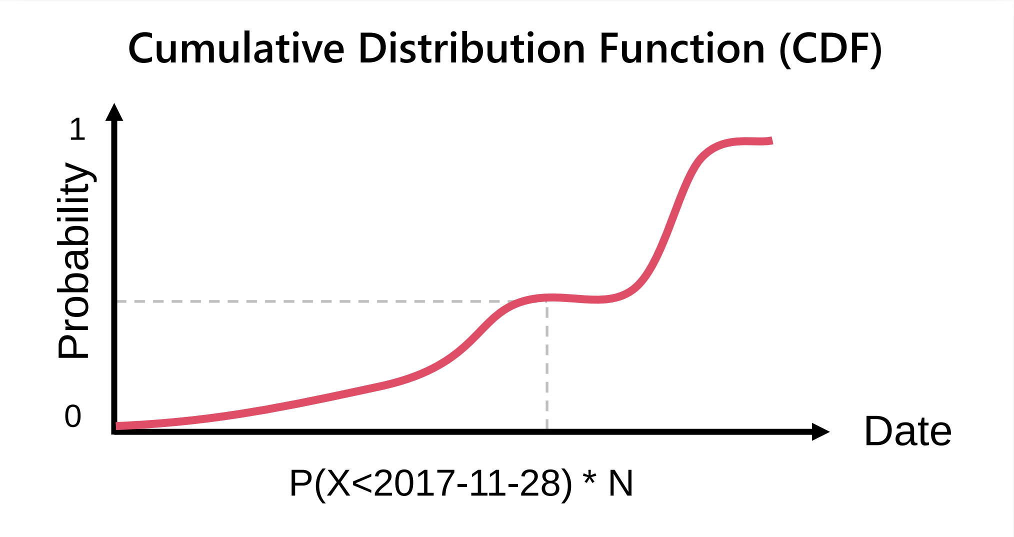


Position



Learned Index

Key
(2017-11-28)



Step 1	Predict Position
Step 2	Localized search to find actual key

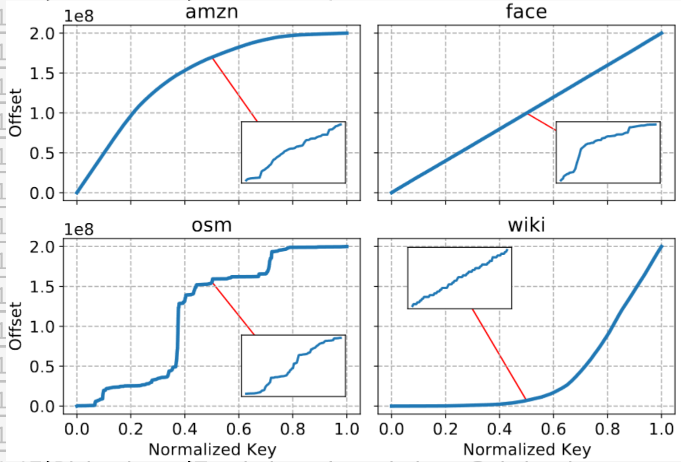
Note: This is NOT an approximate structure
Provides the same guarantees as a B-Tree

Step 1

Step 2

Exploiting the 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				1 Badeau Place				$ 854.00
1010	2017-01				1 Twin Pines Jun				$ 150.00
1011	2017-01				0456 Meadow V				$ 313.00
1012	2017-01				07 Bluestem Jun				$ 558.00
1013	2017-01				7 Fairfield Pass				$ 941.00
1014	2017-01				0995 Manufactu				$ 875.00
1015	2017-01				80 Novick Road				$ 954.00
1016	2017-01				72674 Fuller Ave				$ 239.00
1017	2017-01				87414 Golden Le				$ 308.00
1018	2017-01				2810 Oakridge P				$ 60.00
1019	2017-01				82802 Oakridge				$ 95.00
1020	2017-01				89 Continental A				$ 137.00
1021	2017-01				31 Mcbride Point				$ 496.00
1022	2017-01-17	Rickard	Trenholm	rtrenholmm@cbslocal.com	93 Hoepker Parkway	70593	Louisiana	3541-5241-5383-9970	$ 760.00
1023					14 Eliot Lane			80	$ 474.00
1024					204 Huxley Road			20	$ 83.00
1025					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



Various real-world integer/decimal dataset

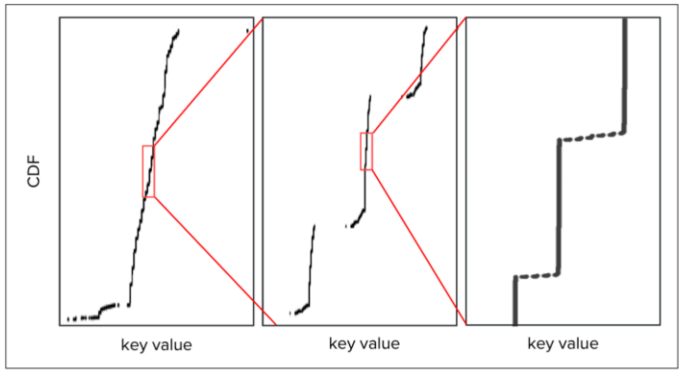
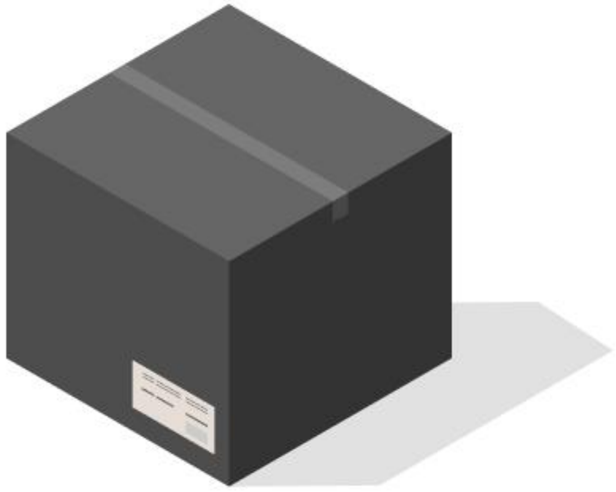


Figure 1-2: Repeated zooming on the CDF of the Wiki dataset reveals its fractally stepwise nature.

Real world string dataset (wiki)

Key Insight of Learned Index



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

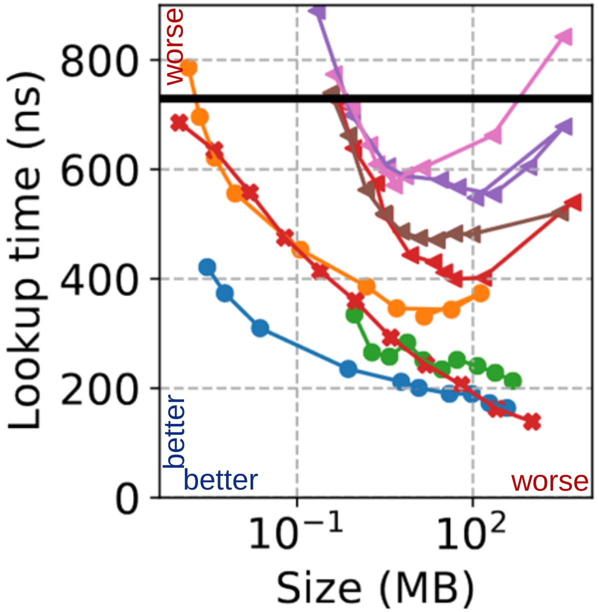
But knowing the data distribution might allow for significant performance gains and might even change the complexity of data structures.

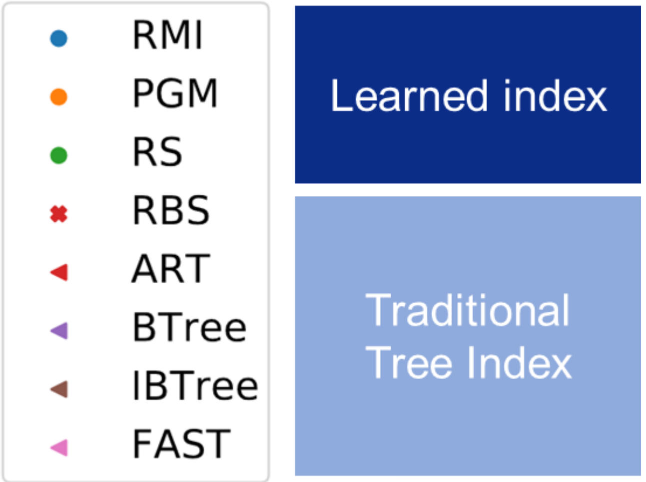
O(log n) → O(1) for lookups
O(n) → O(1) for storage

Ryan Marcus et.al. 2020. Benchmarking learned indexes. Proc. VLDB Endow. 14, 1 (Sep. 2020), 1–13.

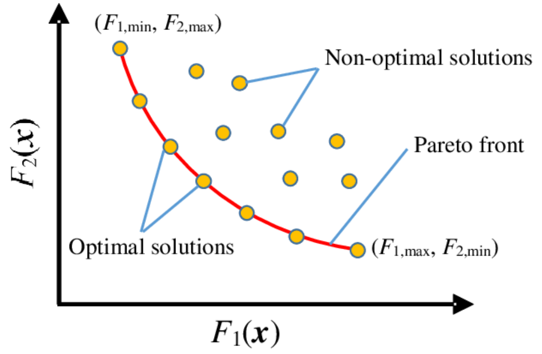
2.5. Performance

- So, how well does learned index perform?





Dataset: Amazon (200M keys)
In-memory database, read-only workload



<*pareto optimal* solutions>

Learned indexes are ***pareto optimal (space-efficient)*** **for size and lookup.**

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.**

Learned Index Structure

- Proposed by Tim Kraska (CSAIL@MIT) in 2018.
  - The Case for Learned Index Structures. (SIGMOD 18).
  - ***Most influential database paper in 2018.***
  - Very active research area

- Work on Learned Indexes appeared in:
  - Database (SIGMOD, VLDB, ICDE, …)
  - System (OSDI, ASPLOS, FAST, …)
  - Machine Learning (NeurIPS, ICML, …)
  - Computer Theory (Theor. Comput. Sci)
  - Network (NSDI)

Index Structure

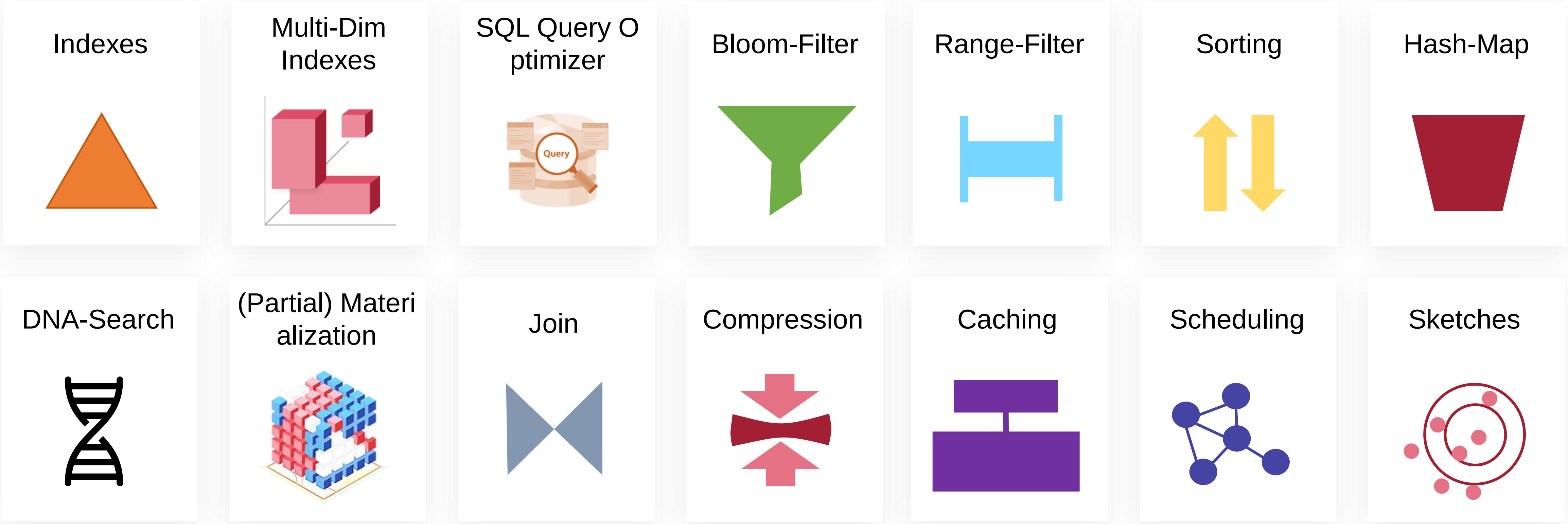
- Learning to hash for indexing big data - A survey (2016)
- The Case for Learned Index Structures (SIGMOD 2018)
- A-Tree: A Bounded Approximate Index Structure (2017)
- FITing-Tree: A Data-aware Index Structure (SIGMOD 2019)
- Learned Indexes for Dynamic Workloads (2019)
- SOSD: A Benchmark for Learned Indexes (2019)
- Learning Multi-dimensional Indexes (2019)
- ALEX: An Updatable Adaptive Learned Index (SIGMOD 2020)
- Effectively Learning Spatial Indices (VLDB 2020) GitHub Link
- Stable Learned Bloom Filters for Data Streams (VLDB 2020)
- START — Self-Tuning Adaptive Radix Tree (ICDEW 2020)
- Learned Data Structures (2020)
- RadixSpline: a single-pass learned index (aiDM2020)
- The ML-Index: A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries (EDBT 2020)
- The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds (VLDB 2020)
- A Tutorial on Learned Multi-dimensional Indexes (SIGSPATIAL 2020)
- Why Are Learned Indexes So Effective? (ICML 2020)
- Learned Indexes for a Google-scale Disk-based Database (arXiv 2020)
- SIndex: A Scalable Learned Index for
- XIndex: A Scalable Learned Index for
- Tsunami: A Learned Multi-dimension 2021)
- A Lazy Approach for Efficient Index L
- The RLR-Tree: A Reinforcement Learr
- Spatial Interpolation-based Learned
- APEX: A High-Performance Learned I
- RUSLI: Real-time Updatable Spline Le
- PLEX: Towards Practical Learned Inde
- SPRIG: A Learned Spatial Index for Ra
- Benchmarking Learned Indexes (VLD
- Updatable Learned Index with Precis
- The Case for Learned In-Memory Joi
- Bounding the Last Mile: Efficient Lea
- FINEdex: A Fine-grained Learned Ind (VLDB 2022)
- The next 50 Years in Database Indexi (VLDB 2022)
- The Concurrent Learned Indexes for
- TONE: cutting tail-latency in learned
- A Learned Index for Exact Similarity S
- RW-tree: A Learned Workload-aware
- The "AI+R"-tree: An Instance-optimi
- LHI: A Learned Hamming Space Inde
- Entropy Learned Hashing: 10X Faster
- Tuning Hierarchical Learned Indexes
- FLIRT: A Fast Learned Index for Rollin
- Testing the Robustness of Learned In
- The Case for ML-Enhanced High-Din
- A Learned Index for Ex
- PLIN: A Persistent Learned Index for Recovery (VLDB 2023)

- A Data-aware Learned Index Scheme for Efficient Writes (ICPP 2022)
- Frequency Estimation in Data Streams: Learning the Optimal Hashing Scheme (TKDE)
- FILM: A Fully Learned Index for Larger-Than-Memory Databases (VLDB 2023)
- WISK: A Workload-aware Learned Index for Spatial Keyword Queries (arXiv 2023)
- Efficiently Learning Spatial Indices (ICDE 2023)
- Cutting Learned Index into Pieces: An In-depth Inquiry into Updatable Learned Indexes (ICDE 2023)
- DILI: A Distribution-Driven Learned Index (arXiv 2023)
- Learned Index: A Comprehensive Experimental Evaluation (VLDB 2023)
- LMSFC: A Novel Multidimensional Index based on Learned Monotonic Space Filling Curves (Extended Version) (arXiv 2023)
- One stone, two birds: A lightweight multidimensional learned index with cardinality support (arXiv 2023)
- A Simple Yet High-Performing On-disk Learned Index: Can We Have Our Cake and Eat it Too? (aiXiv 2023)
- Fast Partitioned Learned Bloom Filter (arXiv 2023)
- Efficient Index Learning via Model Reuse and Fine-tuning (ICDEW 2023)
- COAX: Correlation-Aware Indexing (ICDEW 2023)
- Learned Index with Dynamic e (openreview 2023)
- Learning to Optimize LSM-trees: Towards A Reinforcement Learning based Key-Value Store for Dynamic Workloads (arXiv 2023)
- SALI: A Scalable Adaptive Learned Index Framework based on Probability Models (SIGMODE 2024)
- Sieve: A Learned Data-Skipping Index for Data Analytics (VLDB 2023)
- Demonstrating Waffle: A Self-driving Grid Index (VLDB Demo 2023)
- Can LSH (Locality-Sensitive Hashing) Be Replaced by Neural Network? (arXiv 2023)
- Workload-aware and Learned Z-Indexes (arXiv 2023)
- AirIndex: Versatile Index Tuning Through Data and Storage (SIGMOD 2024)
- A Fast Learned Key-Value Store for Concurrent and Distributed Systems (TKDE 2023)
- When Learned Indexes Meet Persistent Memory: The Analysis and the Optimization (TKDE 2023)
- PLATON: Top-down R-tree Packing with Learned Partition Policy (PACMMOD 2023)
- ized Sliding Windows
- WIPE: a Write-Optimized Learned Index for Persistent Memory (TACO 2023)

https://github.com/LumingSun/ML4DB-paper-list

Types of Instance-Optimized components

- Instance-Optimized algorithms, data structures, and components



… and more

2. Read-only learned indexes

2.1. Index Definition

- A sorted array D of integer keys
  - Index: find the lower bound position of an arbitrary integer x in D.

- Lookup
  - Prediction
    - estimates the position of x in D as p
  - Correction
    - based on p, nearby keys are searched to find the exact position of x

- Error-Bound
  - Error-bounded property : $\forall k \in D, Error(k) \leq \varepsilon$
  - k exist in correction range (= $[p - \varepsilon, p + \varepsilon]$) → binary search

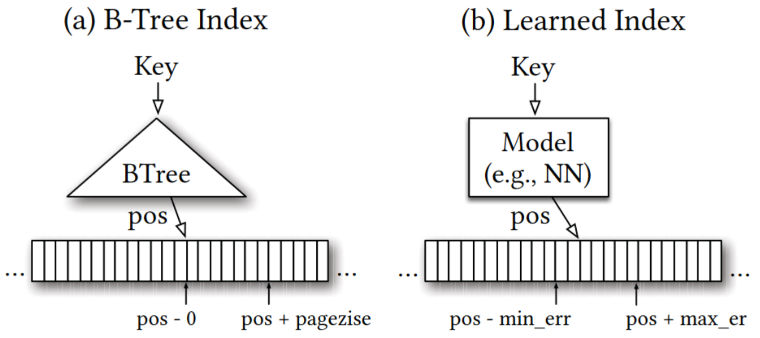
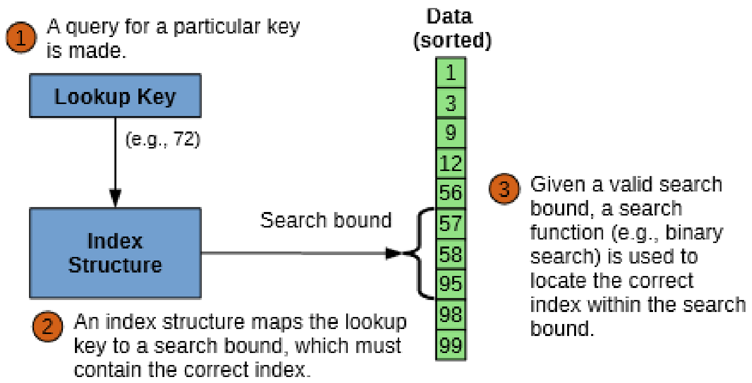
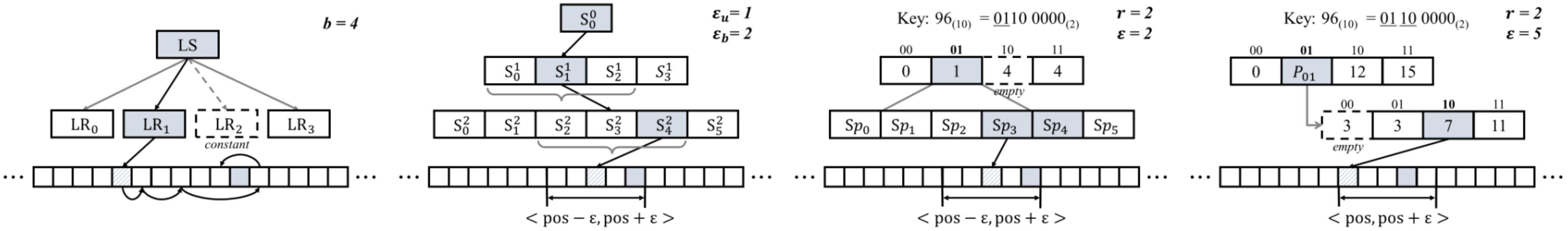


Figure 1: Why B-Trees are models



2.1. Index Definition



(a) RMI (b) PGM-Index (c) RadixSpline (d) Compact Hist-Tree

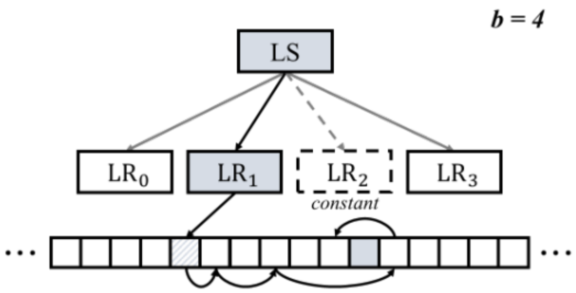
Figure 1: Structure of indexes: RMI, PGM, RS, and CHT.

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.

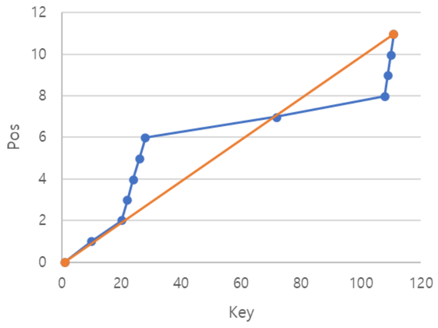
2.2. RMI (Recursive Model Index)

- **Structure**
  - A multi-level learned index
    - supports various sub models.
    - can be configured in various ways

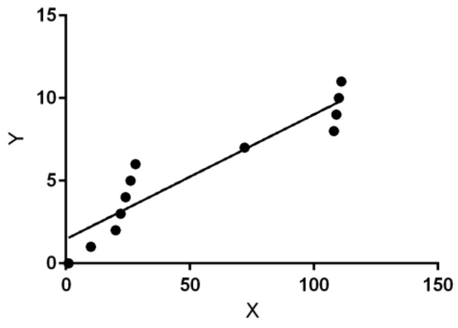


(a) RMI

  - Focus on a two-level RMI:
    - known to be best suited when all keys fit in memory
      - ➢ Top layer : a single linear spline model
      - ➢ Bottom layer : numerous linear regression model
        - ✓ Forms a piecewise linear regression (PLR) model

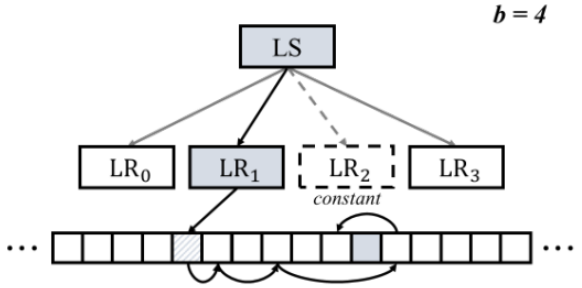


Linear Spline



Linear Regression

2.2. RMI (Recursive Model Index)



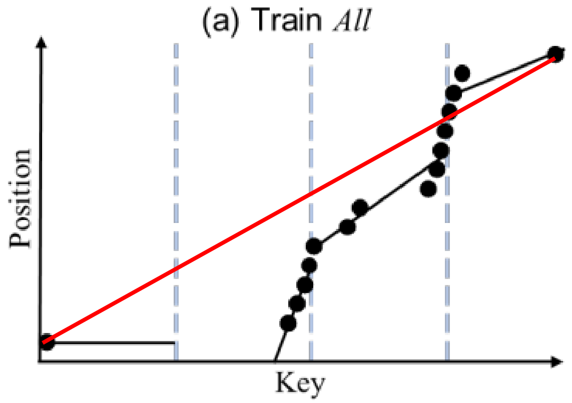
(a) RMI

- **Building**
  - Trains the model from the top layer to the bottom layer.

  - Top Layer (Linear Spline)
    - covers the entire key range using the first and the last key pairs.
    - divides the key range into b number of fixed ranges (i.e., fixed segmentation).
      - b (branching factor) : # of linear segments

  - Bottom Layer (Linear Regression)
    - Each segment is trained to minimize the mean squared error (MSE) of the data within its range.

  - Error Bound
    - Linear regression model itself cannot guarantee the error bound.
    - Should traverse entire dataset again to check each error bound.



Linear Regression Segments in Bottom Layer of RMI

2.2. RMI (Recursive Model Index)

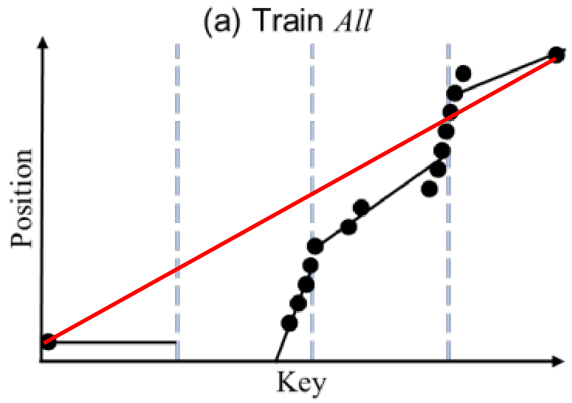
- **Lookup**
  - Prediction
    - Using LS model in top layer,
      - ➢ selects a bottom segment
    - Using chosen linear segment in bottom layer,
      - ➢ predicts the final position of the key in the dataset.
  - Correction
    - If error bound is available
      - ➢ binary search on correction range (error-bound).
    - If not
      - ➢ exponential search based on the predicted position.

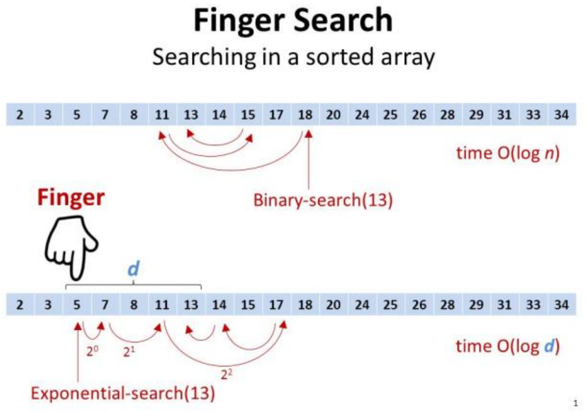
Definition (Prediction): Let R be a k-layer RMI trained on dataset D consisting of $n = |D|$ keys. Let us denote the value p restricted to the interval $[a, b]$ by

$$[\![p]\!]_a^b := \max(a, \min(p, b)). \tag{2}$$

The predicted position for key x of layer l_i is recursively defined as

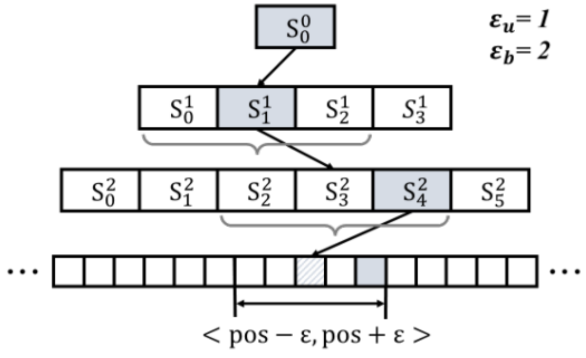
$$f_i(x) = \begin{cases} f_0^0(x) & i = 0 \\ f_i^{\left\lfloor [\![|l_i| \cdot f_{i-1}(x)/n]\!]_0^{|l_i|-1} \right\rfloor}(x) & 0 < i < k \end{cases} \tag{3}$$



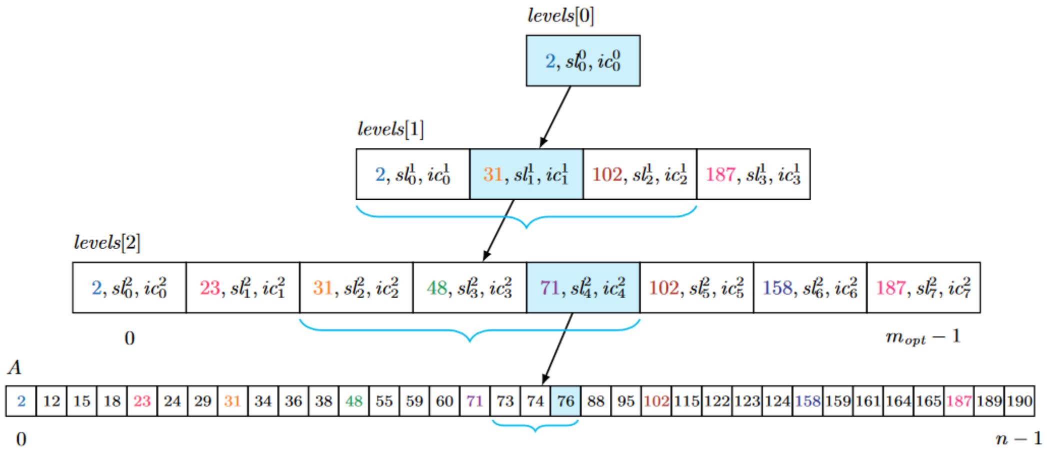


2.3. PGM-Index (piecewise geometric model index)

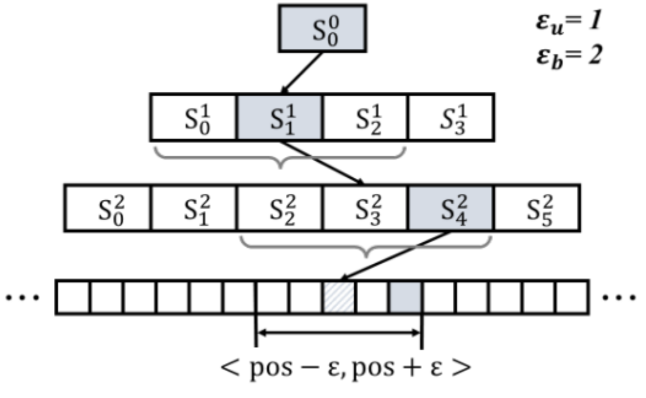
- **Structure**
  - a multi-layered learned index
  - each layer is
    - an error-bounded piecewise linear regression (EB-PLR) model.
  - guarantees a predefined error bound (ε) in all predictions
    - the bottom layer's error bound : ε_{bottom}
    - the upper layers' error bound : ε_{uppper}



(b) PGM-Index



2.3. PGM-Index



(b) PGM-Index

- **Building**
  - a bottom-up approach.
  - Bottom Layer
    - traverses the entire dataset & constructs line segments (EB-PLR).
      - ➢ If an additional data point is out of the given error bound from the current segment,

        → completes the current one → creates an additional line segment on demand

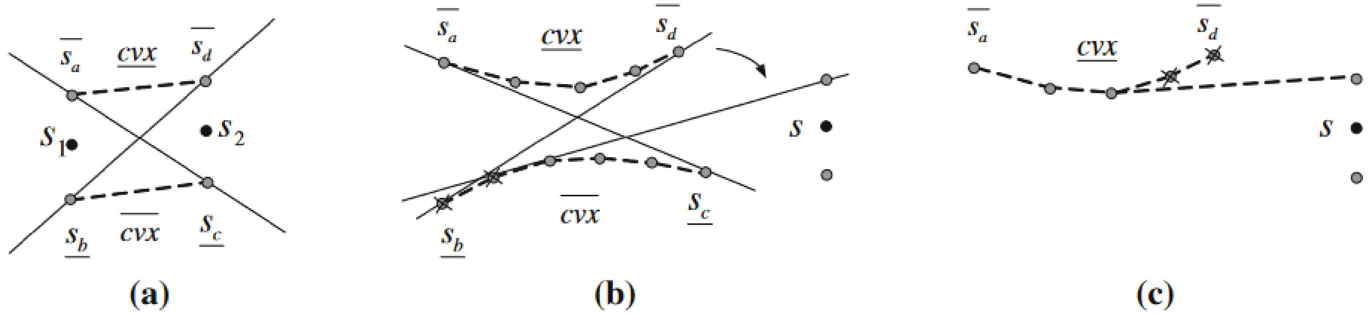
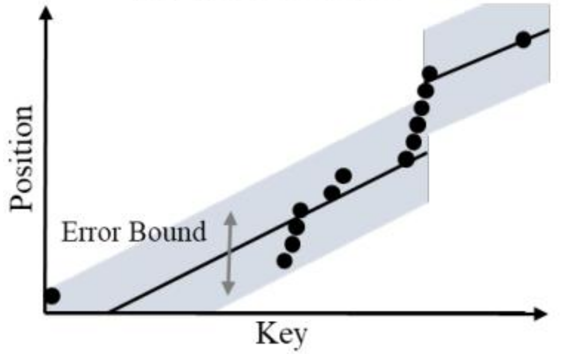
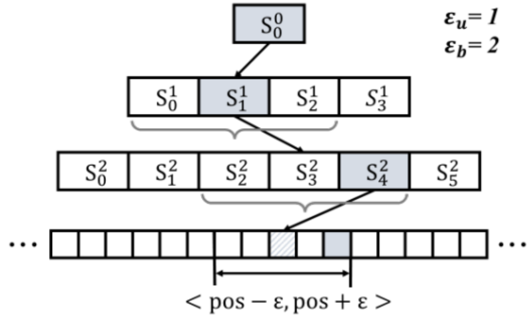


Fig. 7 Optimal algorithm: **a** Initialization; **b** Updating of extreme slopes; **c** Updating of convex hull



Linear Regression Segments in Bottom Layer of PGM

2.3. PGM-Index



(b) PGM-Index

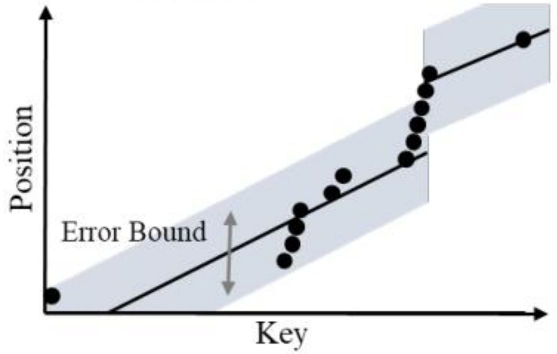
- **Building**
  - a bottom-up approach.

  - Bottom Layer
    - traverses the entire dataset & constructs line segments (EB-PLR).
      - ➢ Each segments approximate the data points within a given error bound.
      - ➢ If an additional data point is out of the given error bound from the current segment,

        → completes the current one → creates an additional line segment on demand

  - Upper Layer
    - traverse segments in lower layer & constructs line segments (EB-PLR).
      - ➢ preserving the error-bound property in its respective layer.
      - ➢ repeats until it reaches the top layer, which has a single segment.

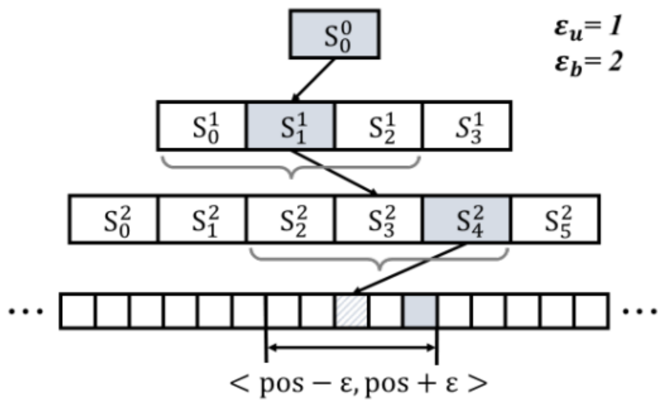
(a) Train *All* with $\varepsilon = 3$



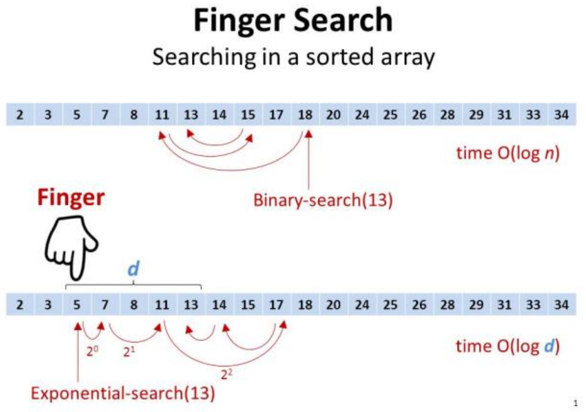
Linear Regression Segments in Bottom Layer of PGM

2.3. PGM-Index

- **Lookup**
  - Prediction
    - top layer,
      - ➢ predicts and finds the segment for the lower layer
      - ➢ until it reaches the bottom layer.
    - bottom layer,
      - ➢ the position of the key in the dataset is predicted.
  - Correction
    - Binary search the key only within the correction range
      - ➢ EB-PLR provides an error-bound for prediction



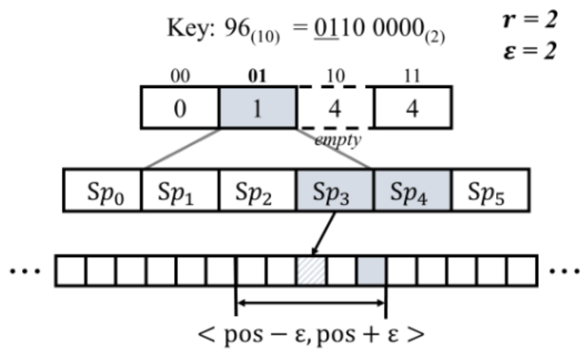
(b) PGM-Index



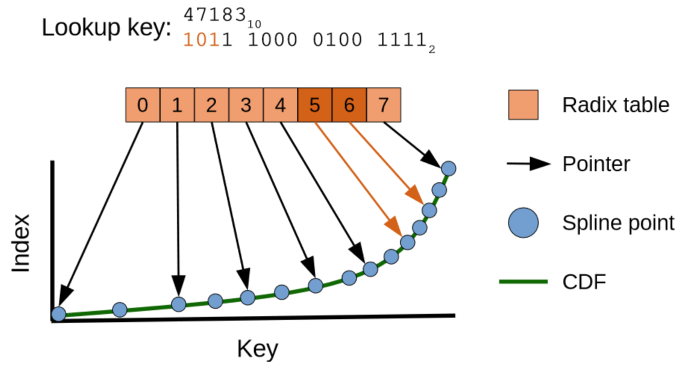
2.4. RS (RadixSpline)

- **Structure**
  - a two-layered learned index
    - The bottom layer : EB-PLS.
      - ➢ error-bounded piecewise linear splines
      - ➢ guarantees an error-bound(ε) for the prediction.

    - The top layer : radix table
      - ➢ A histogram node indexes bottom linear splines
      - ➢ divides the key range into a fixed number of 2^r bins with equal width
        - ✓ r : (prefix) radix bits

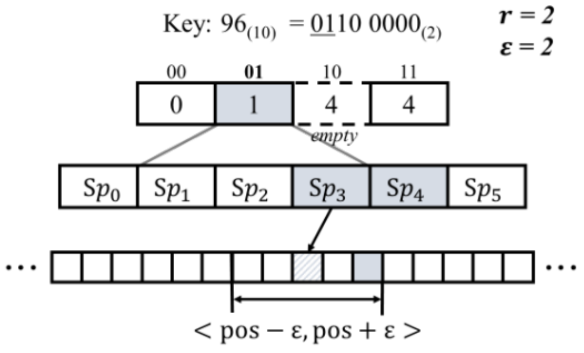


(c) RadixSpline



2.4. RS (RadixSpline)

- **Building**
  - bottom-up.
    - traversing the dataset, constructs the EB-PLS.
    - dynamically segments splines
      - ➢ added data point is out of the given error bound → completes the current spline → begins with a new spline. → the radix table in the top layer is updated accordingly



(c) RadixSpline

GreedySplineCorridor
Input: a spline S, $|S| = n$ and an error corridor size ϵ
Output: a spline connecting $S[1], S[n]$ through the corridor
$B = S[1], R =< B >$ // $S[1]$ is the first base point
$U = S[2] + \epsilon, L = S[2] - \epsilon$ // error corridor bounds
for $i = 3$ **to** n
  $C = S[i]$
  if $\overline{BC}$ is left of $\overline{BU}$ or right of $\overline{BL}$
    $B = S[i-1], R = R\circ < B >$
    $U = C + \epsilon, L = C - \epsilon$
  else
    $U' = C + \epsilon, L' = C - \epsilon$
    if $\overline{BU}$ is left of $\overline{BU'}$
      $U = U'$
    if $\overline{BL}$ is right of $\overline{BL'}$
      $L = L'$
$R = R\circ < S[n] >$
return R

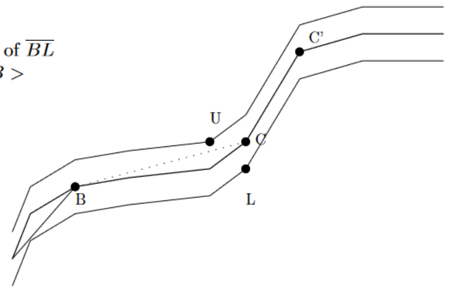
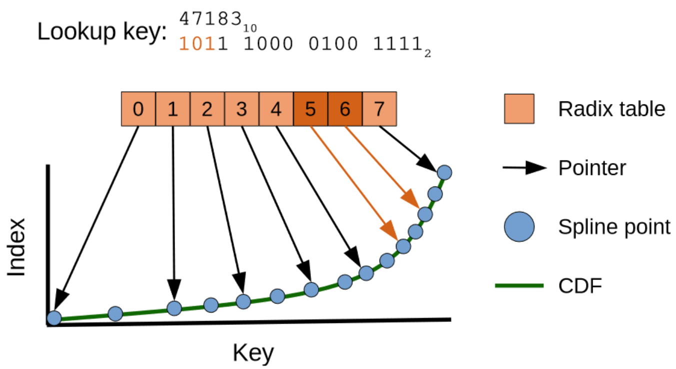
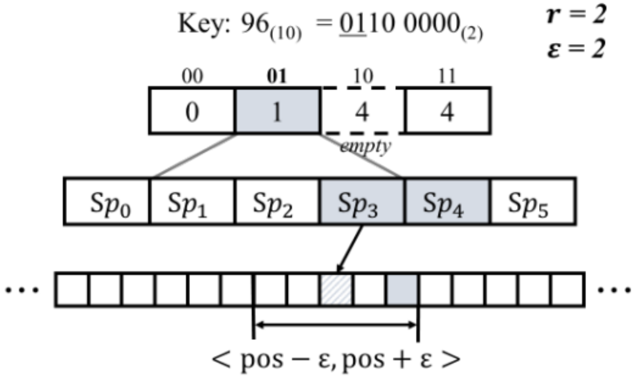


Fig. 1. Greedy Spline Approximation with a Given Error Corridor

2.4. RS (RadixSpline)



- **Lookup**
  - Prediction
    - Radix Table (Top)
      - ➢ computes the lookup key's prefix value to directly access the corresponding bin
      - ➢ probes the positions between the smallest spline in that bin and the subsequent bin
    - EB-PLS (Bottom)
      - ➢ pinpointing two contiguous splines surrounding the lookup key.
      - ➢ the key's position is predicted through spline interpolation.
  - Correction
    - Binary search the key only within the correction range
      - ➢ EB-PLS also provides an error-bound for prediction

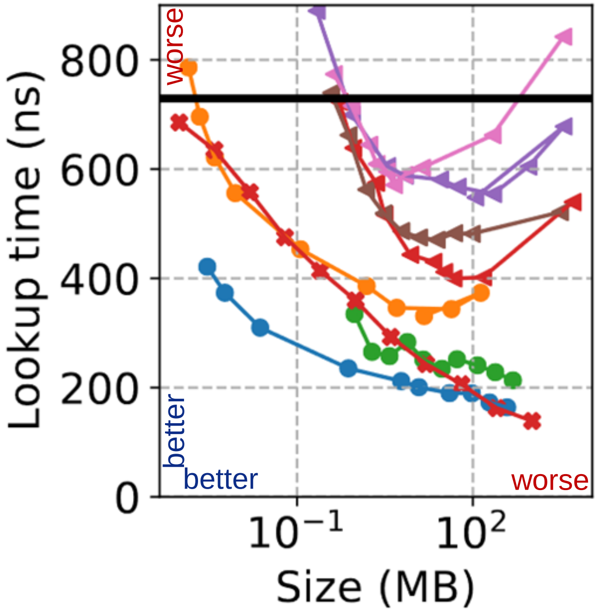


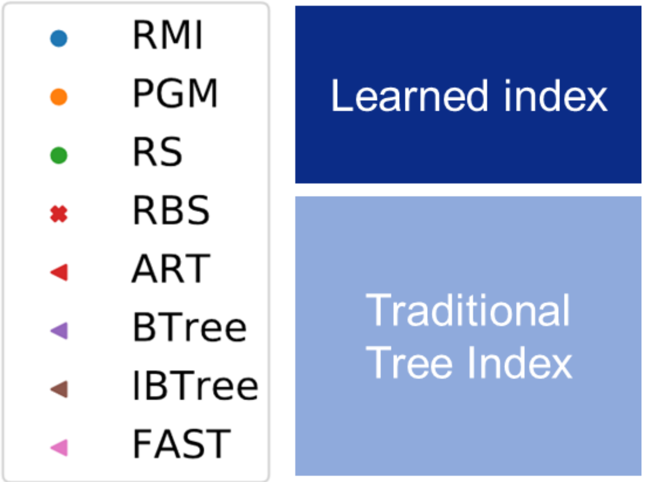
(c) RadixSpline

2.5. Performance

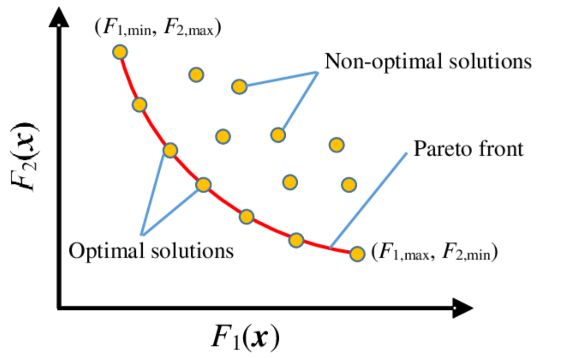
Ryan Marcus et.al. 2020. Benchmarking learned indexes. Proc. VLDB Endow. 14, 1 (Sep. 2020), 1–13.

- So, how well does learned index perform?





Dataset: Amazon (200M keys)
In-memory database, read-only workload



<*pareto optimal* solutions>

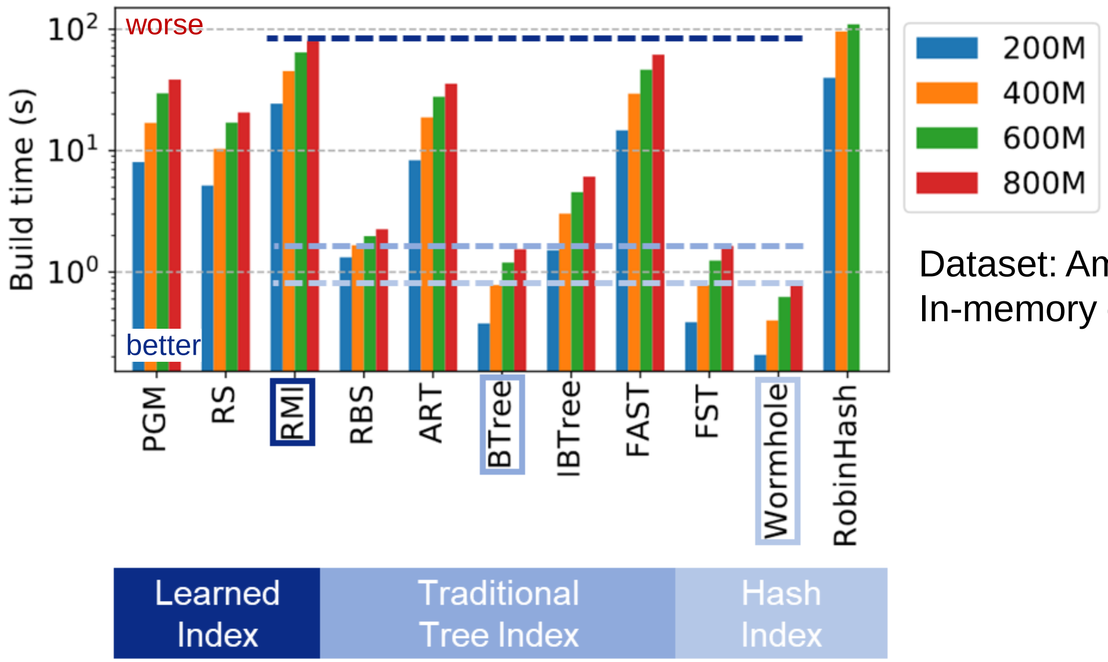
Learned indexes are ***pareto optimal (space-efficient)*** **for size and lookup.**

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.**

2.6. Limitation

Ryan Marcus et.al. 2020. Benchmarking learned indexes. Proc. VLDB Endow. 14, 1 (Sep. 2020), 1–13.

But, learned index is not practical yet.



Dataset: Amazon (# of keys ↑)
In-memory database, read-only workload

Index up to 10x slower in build-time

Learned index requires a complete traversal of the entire dataset
Performance of key-value store suffers on dynamic workload due to long index build-time

3. Applying sampling for learned indexes

Santiago
Chile

SIGMOD
PODS
2024

한국정보과학회 선정 최우수 국제 학술대회
한국연구재단 선정 우수 국제 학술대회 (인정 IF 4)
The ACM Special Interest Group on Management of Data

[SIGMOD' 24]
Can Learned Index be Build-Efficient? A Deep Dive into Sampling Trade-Offs

Minguk Choi[†]　　Seehwan Yoo[‡]　　Jongmoo Choi[†]

School of Software Science[†], School of Mobile System Engineering[‡]
Dankook University

4th Round 1st Review

Accept through shepherding:	***1.4% (4/287)***
Major Revision:	28.2% (81/287)
Reject & withdraw:	70.3% (202/287)

3.1. Applying Sampling on Learned Index

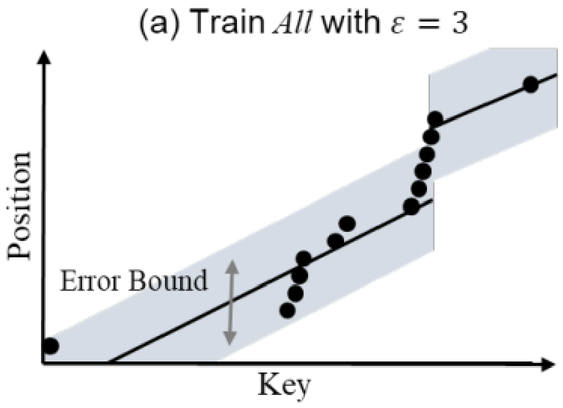
- Sampling may seem too simple, obvious, and even naive.
  - But it poses a series of intricate challenges.

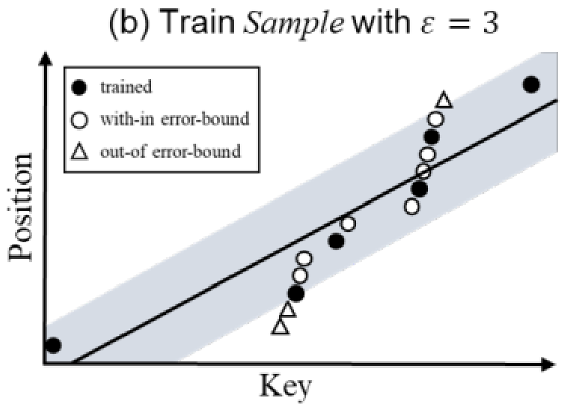
- 3 Challenges
  - 1. Preserving Error-Bound Property
    - Model can not guarantee the error bound for not trained keys.
  - 2. Complex Trade-offs
    - Model (train/size/accuracy), Index (build/size/lookup), Micro-architecture (cache, branch, instructions)
  - 3. Benchmark
    - Fair comparison under unified sampling method and implementations.

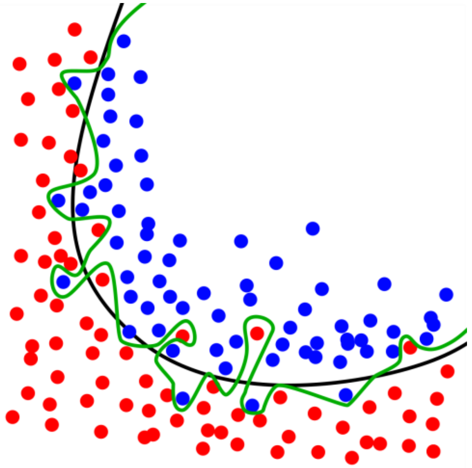
3.2. Error-Bound Property

- Error-Bound Property
  - $For\ \forall\ k \in D,\ Error(k) = |Pred(k) - Pos(k)| \le \varepsilon$
  - important for robustness, especially where correction is expensive. (e.g., disk/remote I/O)

- EB-PLA (Error-Bounded Piece-wise Linear Approximation) Model
  - Learn all the data, dynamically form a segments, ensure all errors are less than ε.
  - Beyond significant overfitting, even limits the maximum error.
  - Cannot guarantee an error-bound for not trained data.

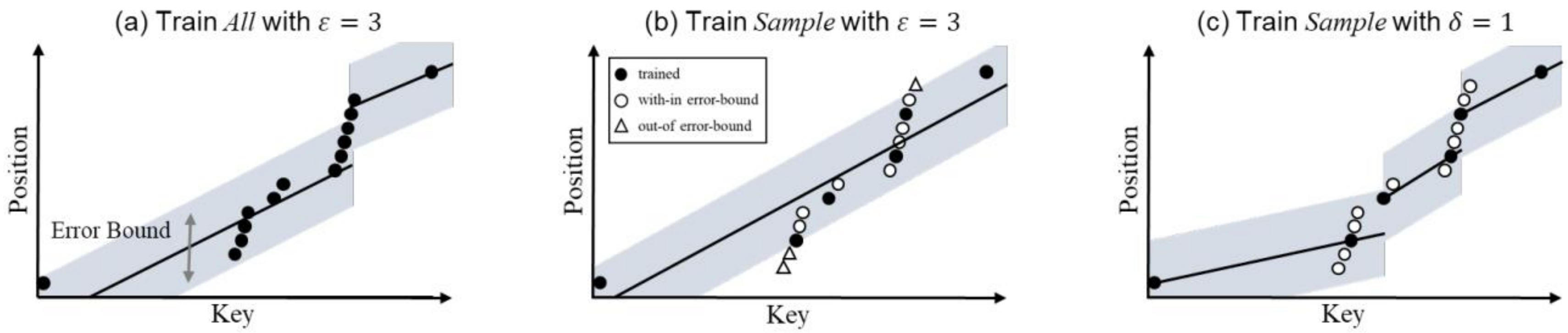






3.2. Error-Bound Property

- Sample EB-PLA Model
  - $For\ \forall\ s \in S,\ Error(s) \leq \delta \rightarrow\ For\ \forall\ k \in D,\ Error(k) \leq \varepsilon\ (= \delta + I - 1)$
  - Guarantee smaller error bound $(= \delta\ = \varepsilon - I + 1)$ for systematic (equal-distance) samples.
  - Aggressive sampling can increase "# of linear segments" (model size).



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

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

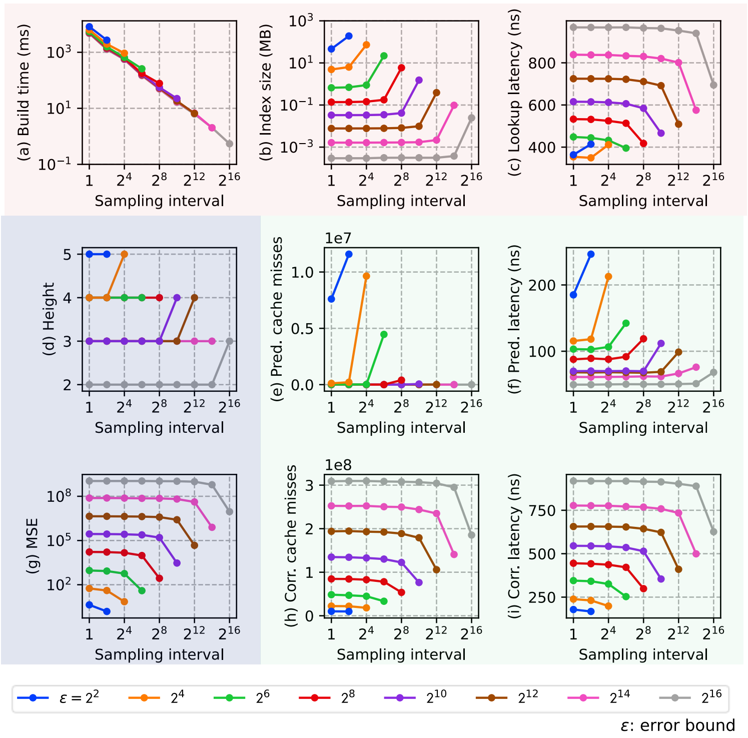
$\square$

3.3. Sampling Trade-offs

Evaluation: Changes in **12 different metrics** (y-axis) depending on the Sampling Interval (x-axis).

1. Index performance
   - (a) Index build time
   - (b) Index size
   - (c) Index lookup latency

2. Internal structure & model
   - (d) Index height
   - (g) Prediction accuracy



3. Micro-architecture
   - (e) Prediction cache miss
   - (f) Prediction latency
   - (h) Correction cache miss
   - (i) Correction latency

Correlation	Instructions	Branch misses	TLB misses	LLC misses
Pred. Lat.	0.987	0.733	0.910	**0.990**
Corr. Lat.	0.986	0.993	0.922	**0.995**

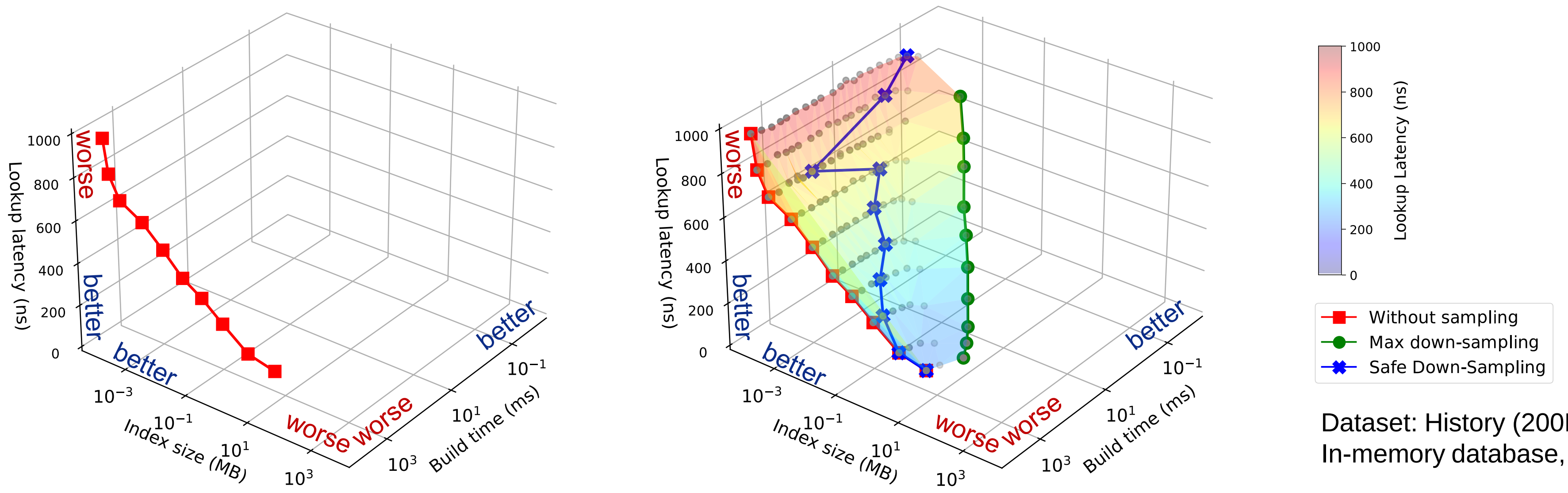
Table 2: Correlation coefficient between the performance counters with lookup latency.

Dataset: History (200M keys)
In-memory database, read-only workload

As the sampling interval increases, the build time decreases effectively (from 8.2 s to 543 μs).
However, aggressive sampling can increase the index size and lookup latency.

3.4. Broaden the Design Space

Incur significant build times regardless of size and lookup → **an absence of trade-offs**.

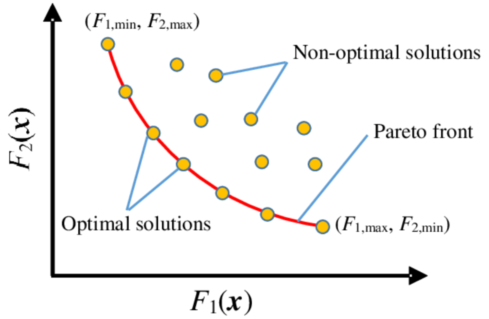


Dataset: History (200M keys)
In-memory database, read-only workload

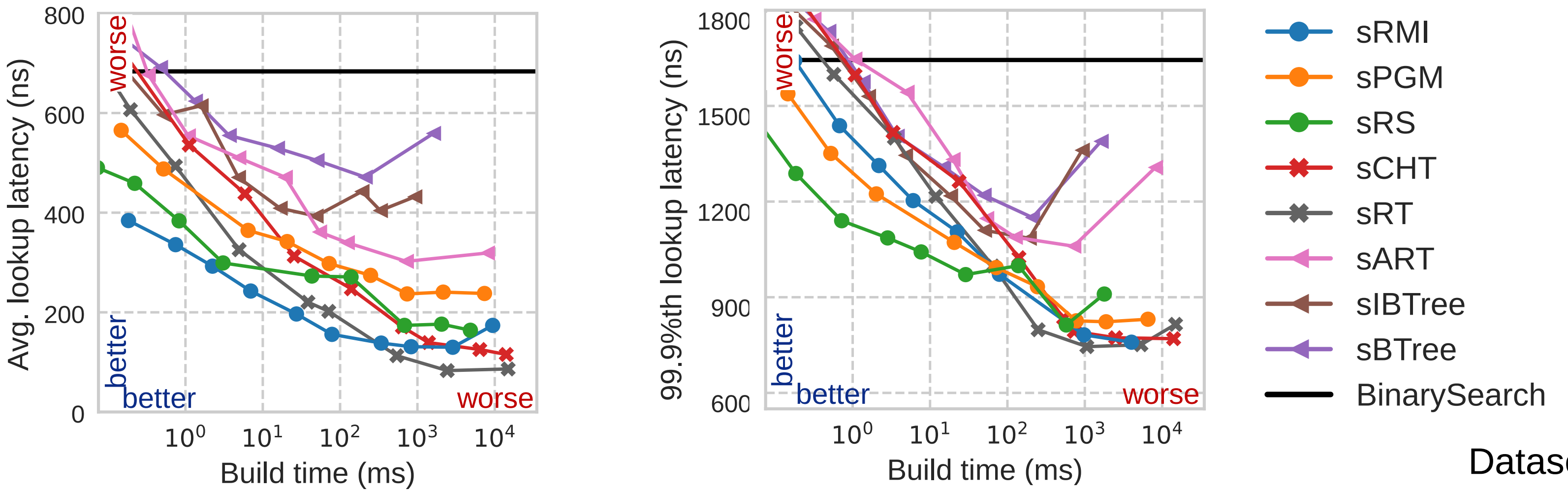
Sampling introduces a trade-off between build/size/lookup → **broadening the design space**.

Reduce build time up to 1/40,781, with an additional 5% size & lookup cost.

3.5. Can Learned Index be Build-efficient?



<*pareto optimal* solutions>



Learned index

Traditional Tree Index

Dataset: Amazon (200M keys)
In-memory database, read-only workload

To the best of our knowledge, it is first to show that
learned indexes are ***pareto optimal (build-efficient)*** **for build and lookup.**

When average latency is 300ns, learned indexes are up to **320x faster in build.**

When tail latency is 1100ns, learned indexes are up to **174x faster in build.**

4. Updatable learned indexes

Various Learned Indexes

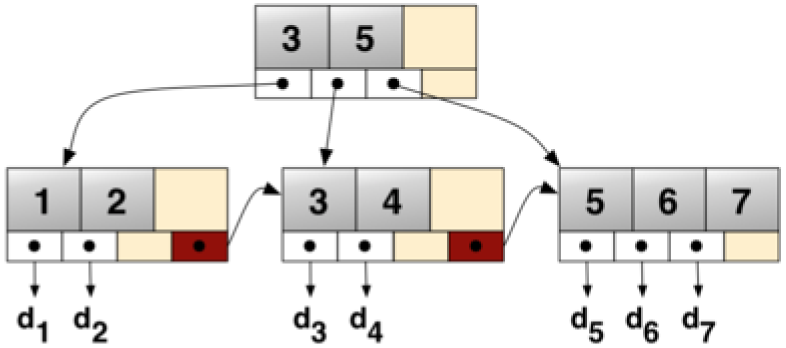
Updatable
Learned
Index

Table 1: Technical differences of evaluated indexes. The thick line separates immutable (top) and mutable indexes (down).

	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
Learned	PLEX [41]	No	No	Non-linear model [4] Linear interpolation	At all nodes	No	Greedy split Bottom-up
Learned	PGM [11]	No	No	Linear model	At all nodes	No	Greedy split Bottom-up
Learned	DPGM (Dynamic PGM [11])	Delta-buffer	Buffer merge [35]	Linear model	At all nodes	No	Greedy split Bottom-up
Learned	XIndex [43]	Delta-buffer	Buffer merge Error-based node split	RMI Piecewise linear regression	At all nodes	Temporary buffer	Even split Bottom-up
Learned	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
Learned	SIndex [46]	Delta-buffer	Buffer merge	(Piecewise) linear regression	At all nodes	Temporary buffer	Greedy split Bottom-up
Learned	ALEX [7]	In-place	Fullness&cost based node expand/split/rebuild	Linear model	At leaf nodes	No	Cost-based split Top-down
Learned	MAB+tree [1]	In-place	Fullness-based node split	Linear interpolation	At all nodes	No	Greedy split Bottom-up
Learned	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
Traditional	ART [20]	In-place	Fullness&prefix based node expand/split	No	At most nodes	No	Top-down
Traditional	B+tree [3]	In-place	Fullness-based node split	No	At all nodes	No	Bottom-up
Traditional	Wormhole [52]	In-place	Fullness-based node split	No	At all nodes	RCU [32] hash table	Bottom-up

ALEX

- B+ Tree
  - Traverses tree using comparisons
  - Supports OLTP-style mixed workloads
    - Point lookups, range queries
    - Inserts, updates, deletes



- RMI
  - Traverses tree using computations (models)
  - Supports point lookups and range queries
  - Advantages : 3x faster reads, 10x smaller size
  - Limitation : does not support writes

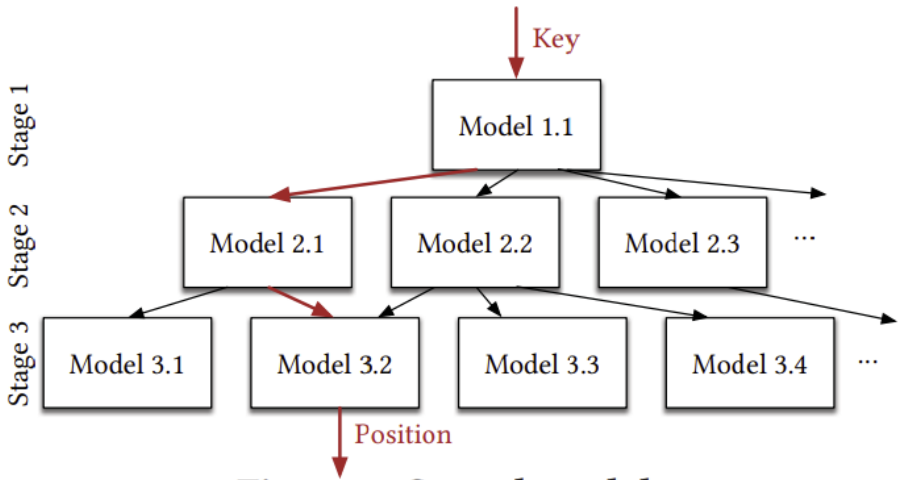


Figure 3: Staged models

ALEX

- Goals
  - Writes competitive with B+ Tree
  - Reads faster than B+ Tree and RMI
  - Index size smaller than B+ Tree and RMI

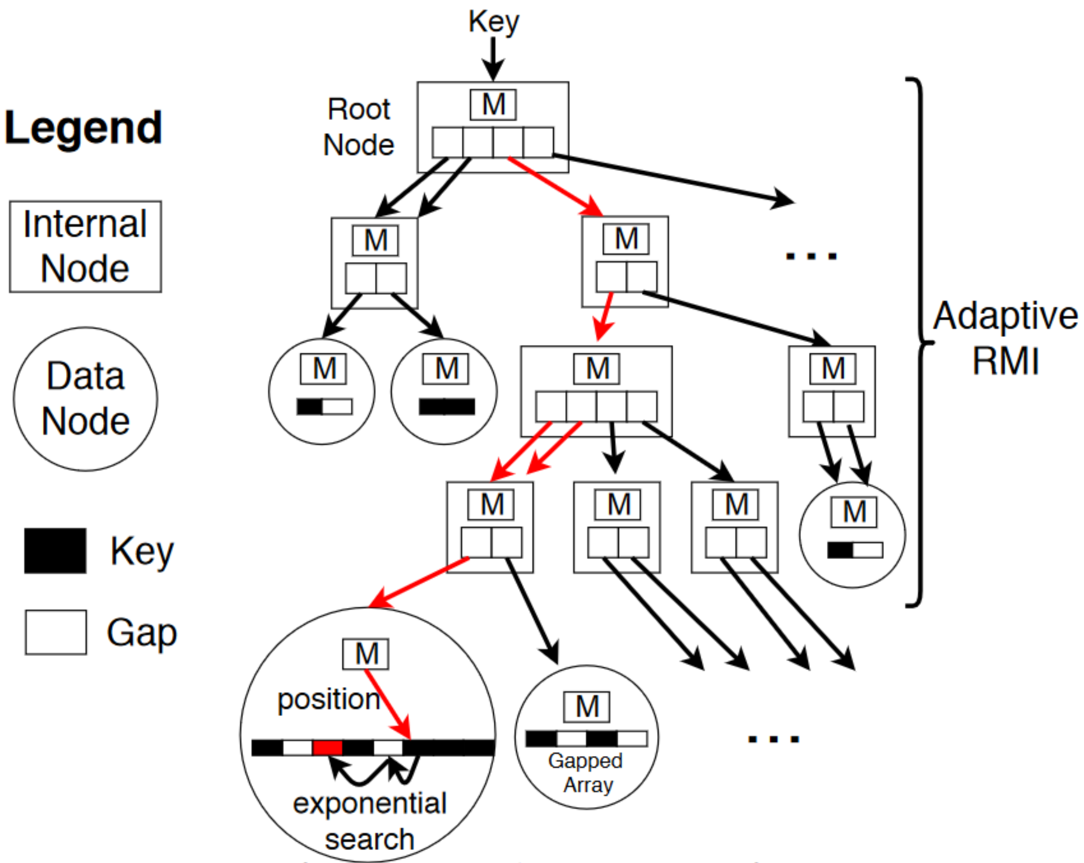


Figure 2: ALEX Design

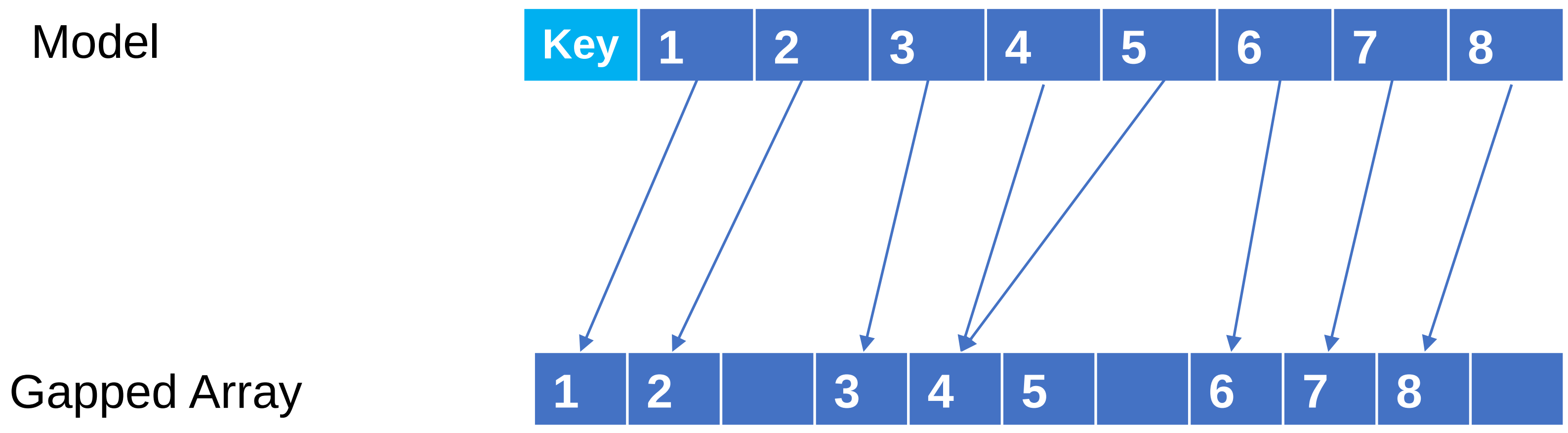
ALEX Core Ideas

	Faster Reads	Faster Writes	Adaptiveness
1. Gapped Array		O	
2. Model-based Inserts	O		
3. Exponential Search	O		
4. Adaptive Tree Structure	O	O	O

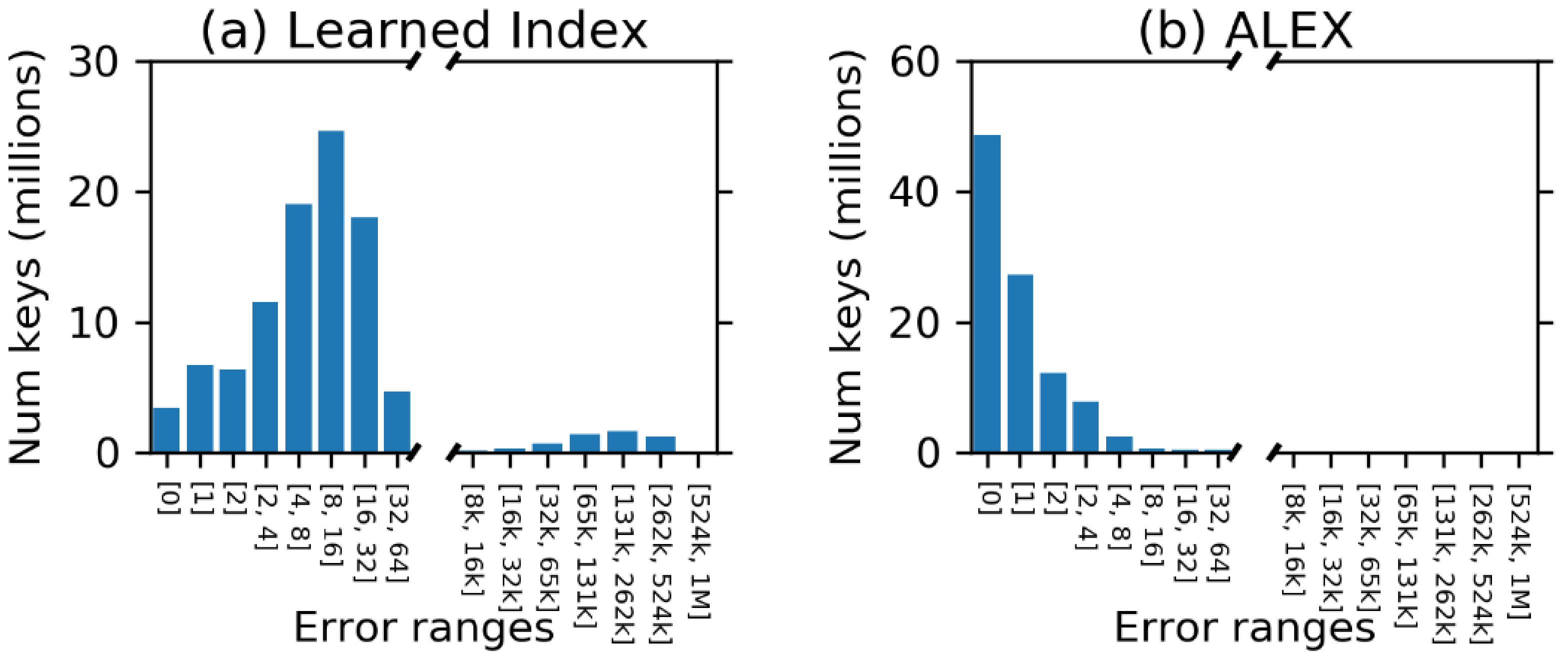
Idea 1. Gapped Array

												Insertion Time
1. Dense Array	0	1	2	3	4	5	6	7	8			O(n)
2. B+ Tree Array	0	1	2	3	4	5	6	7	8			O(n)
3. Gapped Array	0	1	2	3	4		5	6		7	8	O(log n)

Idea 2. Model-based Inserts

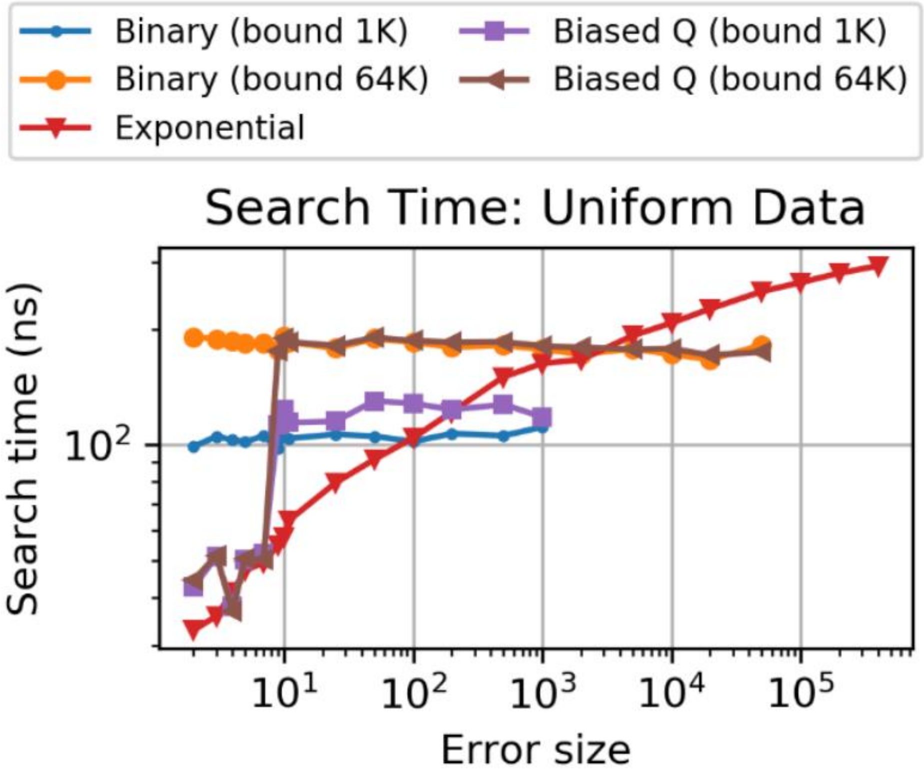


Idea 2. Model-based Inserts



Model-based inserts achieve lower error, leading to faster reads

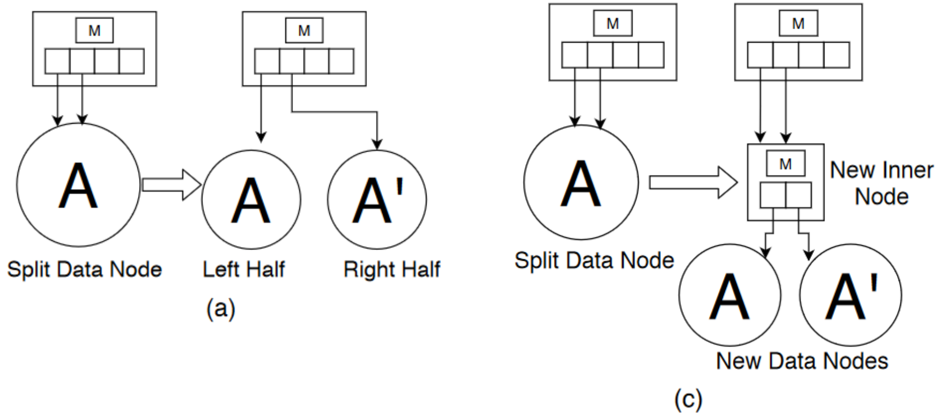
Idea 3. Exponential Search



Model errors are low, so exponential search achieves faster reads

Idea 4. Adaptive Structure

- Alex tree structure
  - Split nodes sideways
  - Split nodes downwards
  - Expand nodes
  - Merge nodes, contract nodes



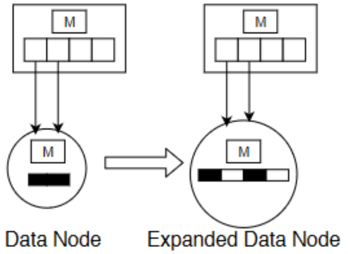


Figure 4: Node Expansion

- Key idea : all decisions are made to maximized performance
  - Use cost model of query runtime
  - No hand-tuning
  - Robust to data and workload shifts

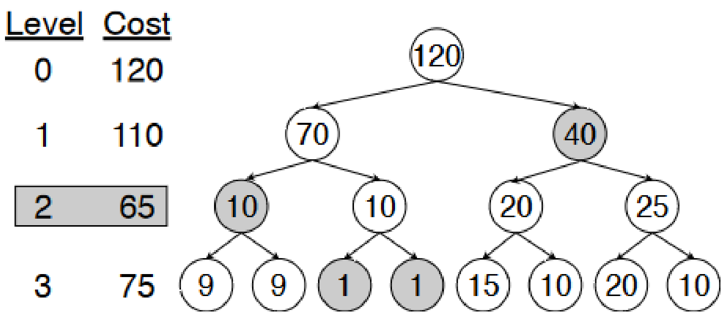
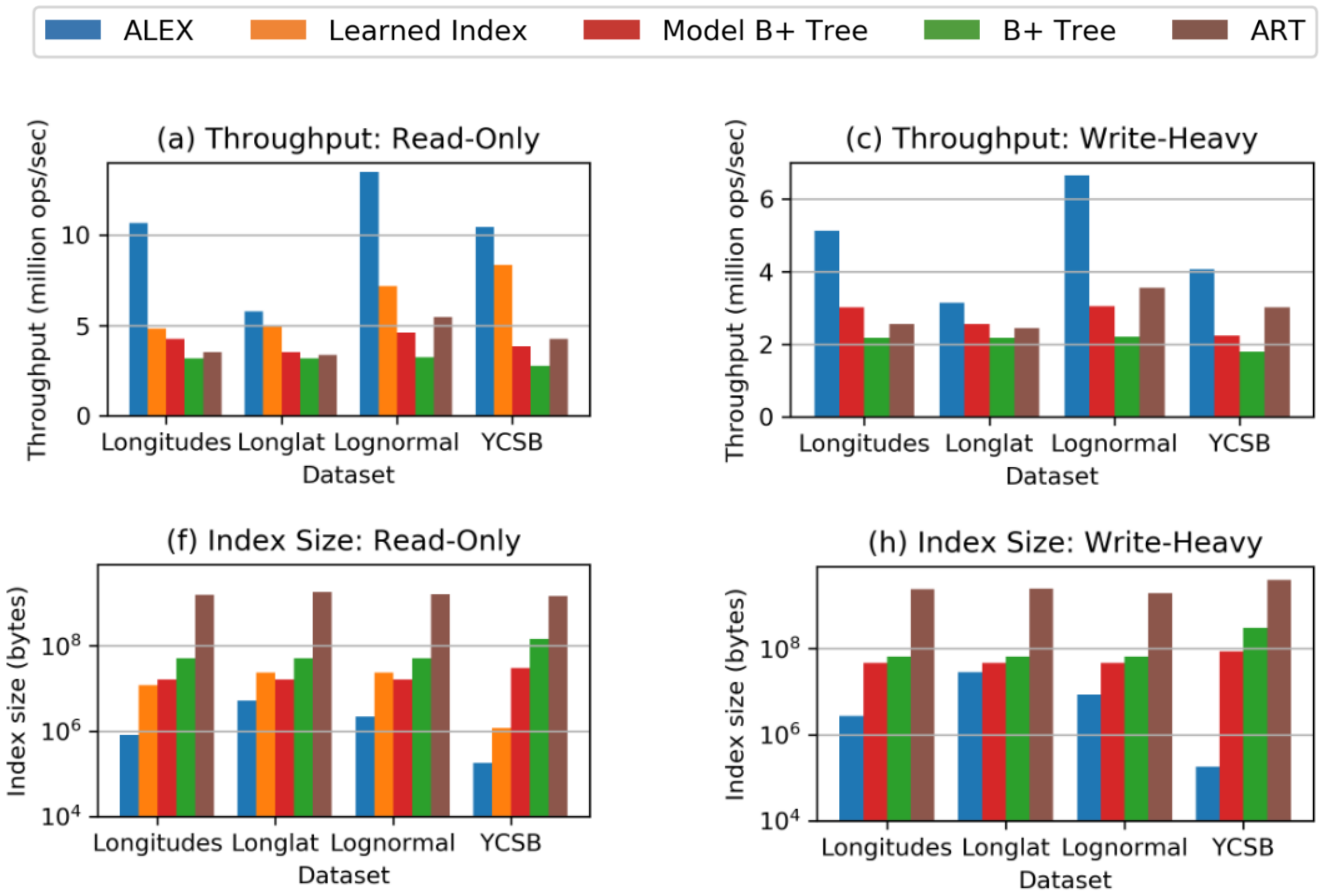


Figure 7: Fanout Tree

Evaluation

- High-level results
  - Fast reads
  - Fast writes
  - Smaller index size
- Other results
  - Efficient bulk loading
  - Scales
  - Robust to data and workload shift



Current Focus & Trends: Scalablity

Table 1: The limited scalability of existing schemes.

Learned	Basic perform.		Concurrency		Evolv.
index	No err	In-place	Fine lock	l-w stat.	ability
RMI[15]	×	×	×	×	×
FITing[8]	×	×	×	×	×
PGM[5]	×	×	×	×	×
ALEX+[3]	×	√	×	√	×
LIPP+[39]	√	√	√	×	×
XIndex[36]	×	×	√	×	×
FINEdex[21]	×	×	√	×	×
SALI	**√**	**√**	**√**	**√**	**√**

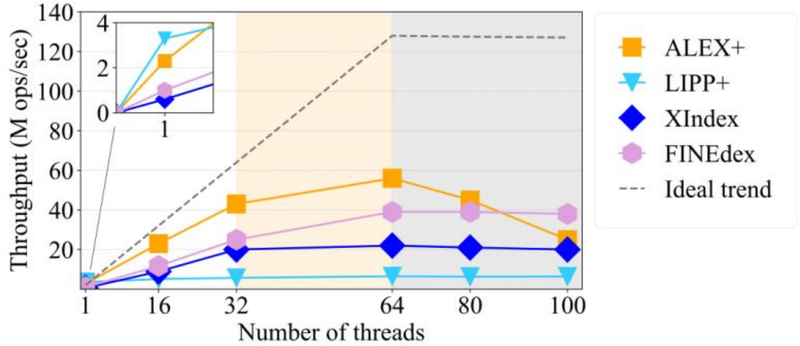


Figure 1: Write-only performance of state-of-the-art learned indexes on the FACE dataset [11]. The evaluation is conducted on a two-socket machine with two 16-core CPUs.

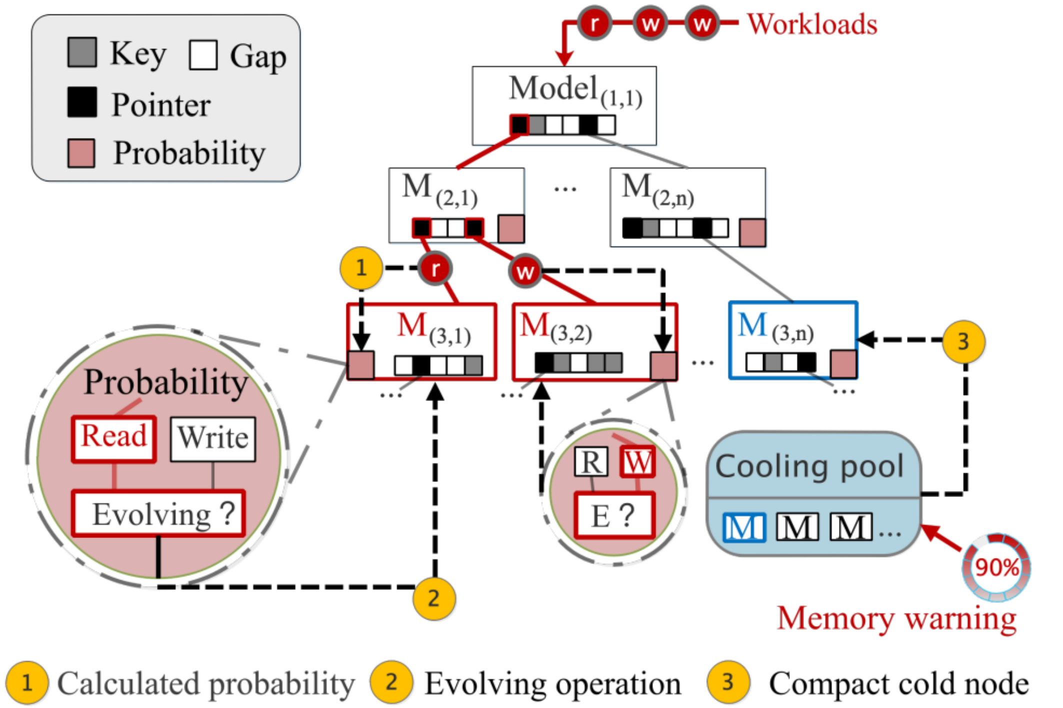


Figure 6: The structure of SALI builds upon the *Mod.* + *C*.

5. Research Topics

Paper Lists & Benchmarks

Learned Index

Read-Only Learned Index

- Maltry, Marcel, et al. "A critical analysis of recursive model indexes.", VLDB 22'
- Ferragina, Paolo, et al. "The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds.", VLDB 20
- Kipf, Andreas, et al. "RadixSpline: a single-pass learned index.", aiDM@SIGMOD 20
- Marcus, Ryan, et al. **"Benchmarking learned indexes."**, VLDB 20
- Minguk, Choi, et al. **"Can Learned Indexes be Build-Efficient? A Deep Dive into Sampling Trade-Offs."**, SIGMOD 24

Updatable Learned Index

- Ding, Jialin, et al. "ALEX: an updatable adaptive learned index.", SIGMOD 20
- Wu, Jiacheng, et al. "Updatable learned index with precise positions.", VLDB 21
- Ge, Jiake, et al. **"SALI: A Scalable Adaptive Learned Index Framework based on Probability Models."**, SIGMOD 23
- Wongkham, Chaichon, et al. **"Are updatable learned indexes ready?"**, VLDB 22'
- Sun, Zhaoyan, et al. **"Learned Index: A Comprehensive Experimental Evaluation."** VLDB 23.

Algorithm & Application

- Error-Bounded PLA Model
  - Xie, Qing, et al. **"Maximum error-bounded piecewise linear representation for online stream approximation."** VLDB journal 14
- Key-Value Store
  - Dai, Yifan, et al. "From {WiscKey} to Bourbon: A Learned Index for {Log-Structured} Merge Trees.", OSDI 20
  - Yu, Geoffrey X., et al. "Treeline: an update-in-place key-value store for modern storage.", VLDB 22
- NVM Device
  - Lu, Baotong, et al. "APEX: A high-performance learned index on persistent memory.", VLDB 21
  - Ge, Jiake, et al. "Cutting Learned Index into Pieces: An In-depth Inquiry into Updatable Learned Indexes." ICDE 23.
- FTL
  - Sun, Jinghan, et al. **"Leaftl: A learning-based flash translation layer for solid-state drives."**, ASPLOS 23

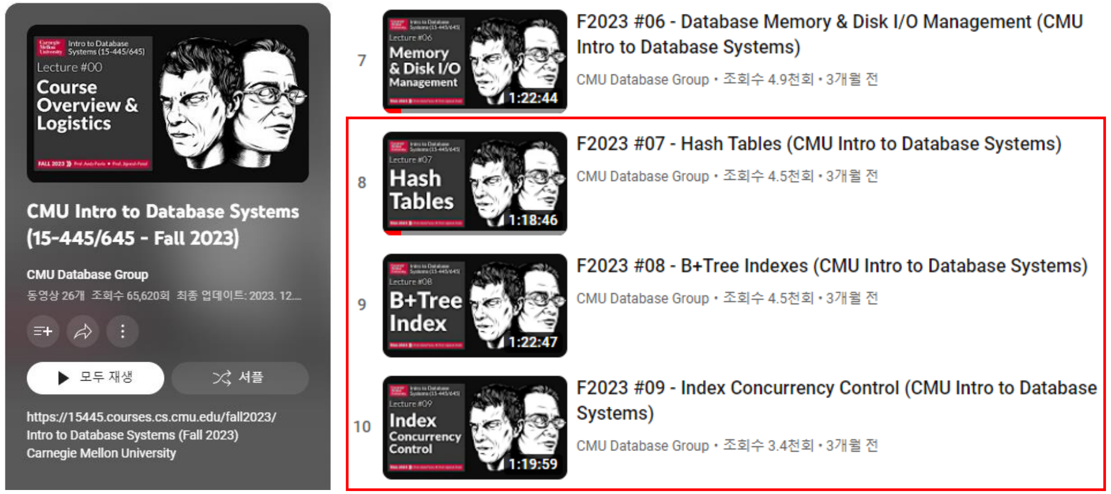
Extensive ML4System Paper List

- LumingSun, "ML4DB-paper-list"

🔗 Benchmarks

- Traditional Index : Index-Microbench
- Read-Only Learned Index : LIST
- Updatable Learned Index : GRE (with SALI)

Lectures



AWS Data Insights Day | AWS On Air ft. How will AI revolutionize databases?

Research Topics

- I'm currently focusing on ***"making learned index practical"***
  - Reducing training overhead which makes learned index impractical
    - (Adaptive) Sampling, SIMD
  - 1. Applying SIMD on RMI & ALEX
    - Use CPU parallelism to reduce training overhead
    - SIMD can be applied only on model without data-dependency
    - There are **no learned index using SIMD yet**.
  - 2. Applying Sampling on EB-PLA based learned index applications
    - Various applications are struggles with training overhead of learned indexes.
    - There are **no applications using sampling applied learned index yet.**
    - E.g., Key-Value Store, Disk-based learned index, Learned-FTL, …
  - 3. Adaptive sample learning algorithm for EB-PLA model
    - Current Sample EB-PLA algorithm asks the user to set the sampling interval.
    - Develop an algorithm that **learns almost identically to the original through the optimal sampling interval**.
    - I'm now recruiting a co-researcher (author) to work with. → Submit to SIGMOD/VLDB 25 !

6. Tips & Tools

6.1. For Beginners

- As a computer system researcher, basic things you should know:
  - Environment Setup & Experimentation: *linux, makefile, vim, vscode*
  - Code Analysis: *gdb, uftrace, understand*
  - Experimentation and Graphing: *shell programming, matplotlib*
  - How to Read Papers: *S. Keshav, "How to Read a Paper"*
  - Mindset :
    - *Remzi Arpaci-Dusseau, "Graduate School: Keys To Success"*
    - *권창현, "박사과정 학생이 유의해야 하는 점"*
    - *문수복, "개별연구/졸업연구를 고민하는 학생들을 위한 간추린 자료"*

6.2. Environment Setup & Experimentation

- Environment Setup & Experimentation: *linux, makefile, vim*

For Beginners

- Lectures
  - SNU 쓰디연구소, JTJ의 리눅스탐험 (linux, makefile, vim, gdb)
  - 따라하면서 배우는 Shell Programming
  - Matplotlib 단기집중과정
  - 제대로 파는 Git & GitHub - 깃 끝.장.내.기
- Documents
  - 권창현, "박사과정 학생이 유의해야 하는 점"
  - 문수복, "개별연구/졸업연구를 고민하는 학생들을 위한 간추린 자료"
  - 문수복, "논문 제 1 저자의 책임"
  - S. Keshav, "How to Read a Paper"
  - Remzi Arpaci-Dusseau, "Graduate School: Keys To Success"
  - Remzi Arpaci-Dusseau, "25 Years of Storage Research and Education: A Retrospective"
- Tools
  - GDB
  - Understand
  - Uftrace
- File
  - presentation file format
  - kcc research paper format
- Previous Study
  - DKU Leveldb Study, 2022
  - DKU RocksDB Festival, 2021

6.3.1. Code Analysis: *understand*

- Understand by SciTools
  - https://www.scitools.com/ (ENG.)
  - https://www.slexn.com/understand/ (KOR.)

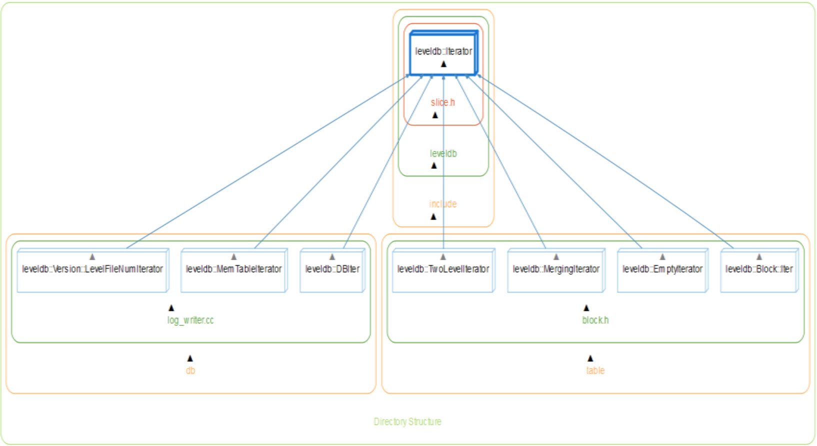
- Static analysis tool
  - understanding of complex open-source code
  - Graphing
    - code dependencies, code flows, function calls, and more

- How to install
  - Register with university email
    - https://licensing.scitools.com/register
  - Free trial link
    - https://www.scitools.com/pricing
  - Download link
    - https://licensing.scitools.com/download

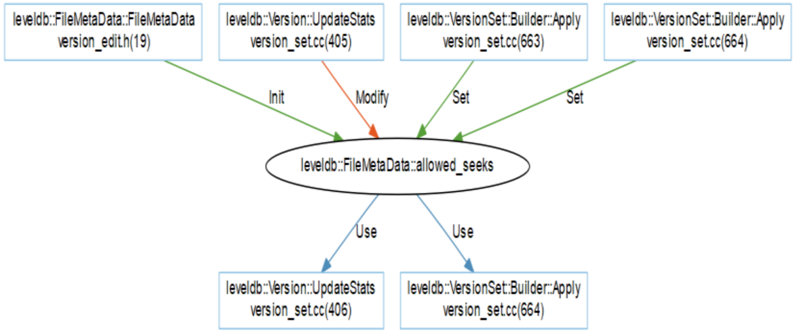
Free for Educational Use

Understand is free for students and teachers to use for educational purposes. Want to teach a class on code maintenance or need some metrics for your thesis? No problem, we got you covered. Learn More

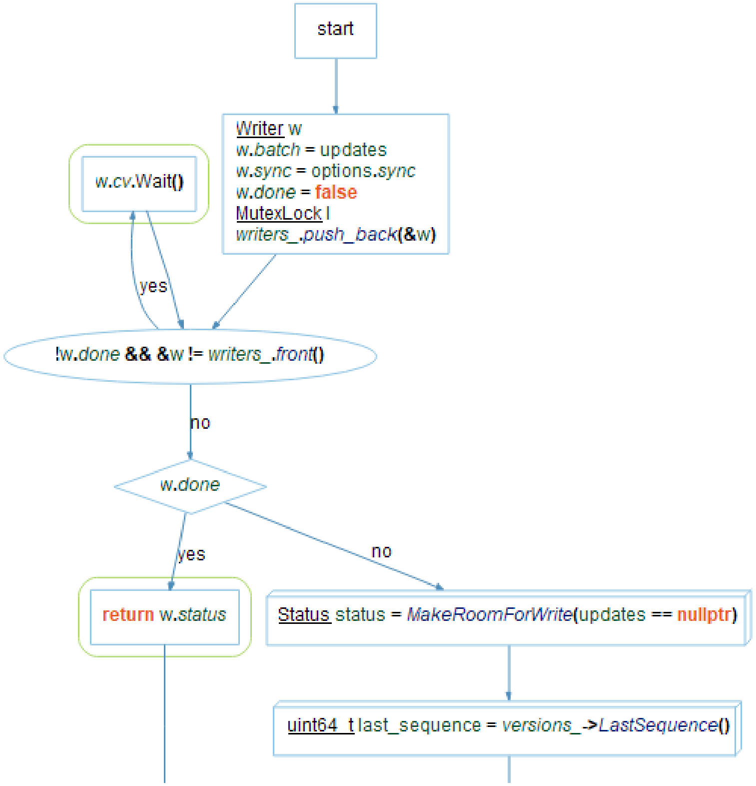
6.3.1. Code Analysis: *understand*



UML Class Diagram: Iterator

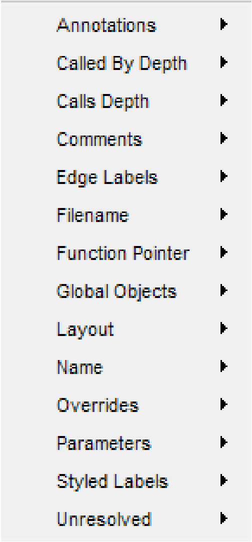


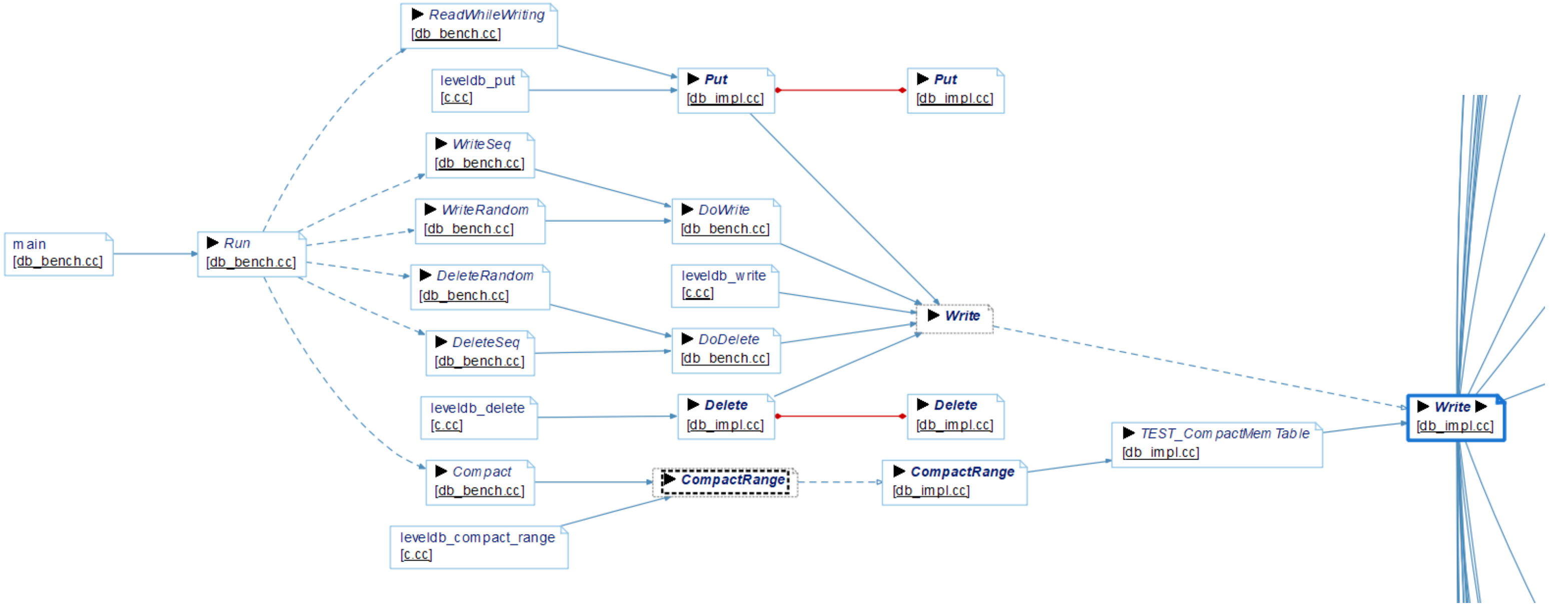
Object References

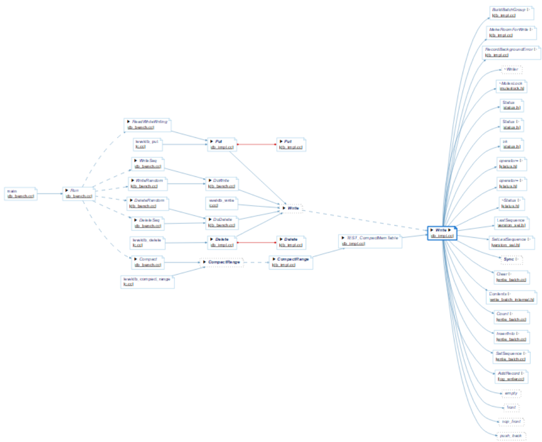


Function Flow

6.3.1. Code Analysis: *understand*







Butterfly: Function Calls & Called by

6.3.2. Code Analysis: *GDB*

- What is GDB?
  - GNU Project debugger
  - see what is going on `inside' another program while it executes
  - see what another program was doing at the moment it crashed.

- Key features
  - Start your program, specifying anything that might affect its behavior.
  - Make your program stop on specified conditions.
  - Examine what has happened, when your program has stopped.
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

GDB
The GNU Project Debugger

GNU

6.3.2. Code Analysis: *GDB*

- LevelDB Example: Merge Iterator

```
1013        input->Next();
(gdb) s
leveldb::(anonymous namespace)::MergingIterator::Next (this=
    0x555555590fee <leveldb::(anonymous namespace)::TwoLevelIterator::value() const+96>)
    at /home/mingu/leveldb_release/table/merger.cc:55
55        void Next() override {
(gdb) n
56          assert(Valid());
(gdb)
63          if (direction_ != kForward) {
(gdb)
77          current_->Next();
(gdb)
78          FindSmallest();
(gdb) s
leveldb::(anonymous namespace)::MergingIterator::FindSmallest (
    this=0x555555590f8c <leveldb::(anonymous namespace)::TwoLevelIterator::key() const+96>)
    at /home/mingu/leveldb_release/table/merger.cc:148
148     void MergingIterator::FindSmallest() {
(gdb)
```

6.3.2. Code Analysis: *GDB*

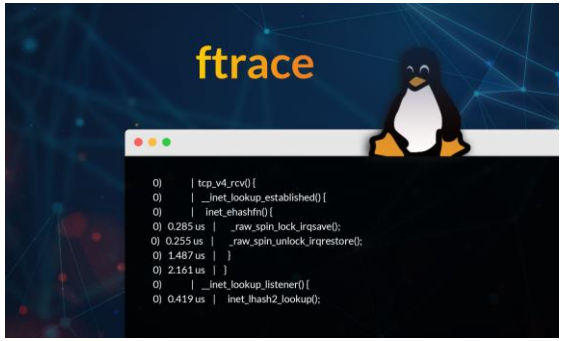
- LevelDB Example: Merge Iterator

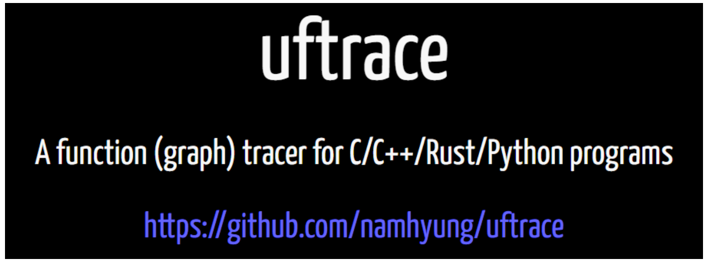
```
(gdb) display smallest
1: smallest = (leveldb::IteratorWrapper *) 0x0
(gdb) display child
2: child = (leveldb::IteratorWrapper *) 0x7fffe8010438
(gdb) n
150         for (int i = 0; i < n_; i++) {
1: smallest = (leveldb::IteratorWrapper *) 0x7fffe8010438
(gdb)
151           IteratorWrapper* child = &children_[i];
1: smallest = (leveldb::IteratorWrapper *) 0x7fffe8010438
2: child = (leveldb::IteratorWrapper *) 0x7fffe8010438
(gdb) display *smallest
3: *smallest = {iter_ = 0x7fffe8012110, valid_ = true, key_ = {
    data_ = 0x7fffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
(gdb) display *child
4: *child = {iter_ = 0x7fffe8012110, valid_ = true, key_ = {
    data_ = 0x7fffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
(gdb) n
152           if (child->Valid()) {
1: smallest = (leveldb::IteratorWrapper *) 0x7fffe8010438
2: child = (leveldb::IteratorWrapper *) 0x7fffe8010458
3: *smallest = {iter_ = 0x7fffe8012110, valid_ = true, key_ = {
    data_ = 0x7fffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
4: *child = {iter_ = 0x7fffe800ff50, valid_ = true, key_ = {
    data_ = 0x7fffe8012e30 '0' <repeats 14 times>, "40\001\213\206\001", size_ = 24}}
(gdb)
```

```
void MergingIterator::FindSmallest() {
  IteratorWrapper* smallest = nullptr;
  for (int i = 0; i < n_; i++) {
    IteratorWrapper* child = &children_[i];
    if (child->Valid()) {
      if (smallest == nullptr) {
        smallest = child;
      } else if (comparator_->Compare(child->key(),
        smallest = child;
      }
    }
  }
  current_ = smallest;
}
```

6.3.3. Code Analysis: Uftrace

- To trace and analyze execution of a program written in C/C++
- Heavily inspired by the ftrace framework of the Linux kernel
- Supports user-space programs and kernel
- Various kind of commands and filters





6.3.3. Code Analysis: Uftrace

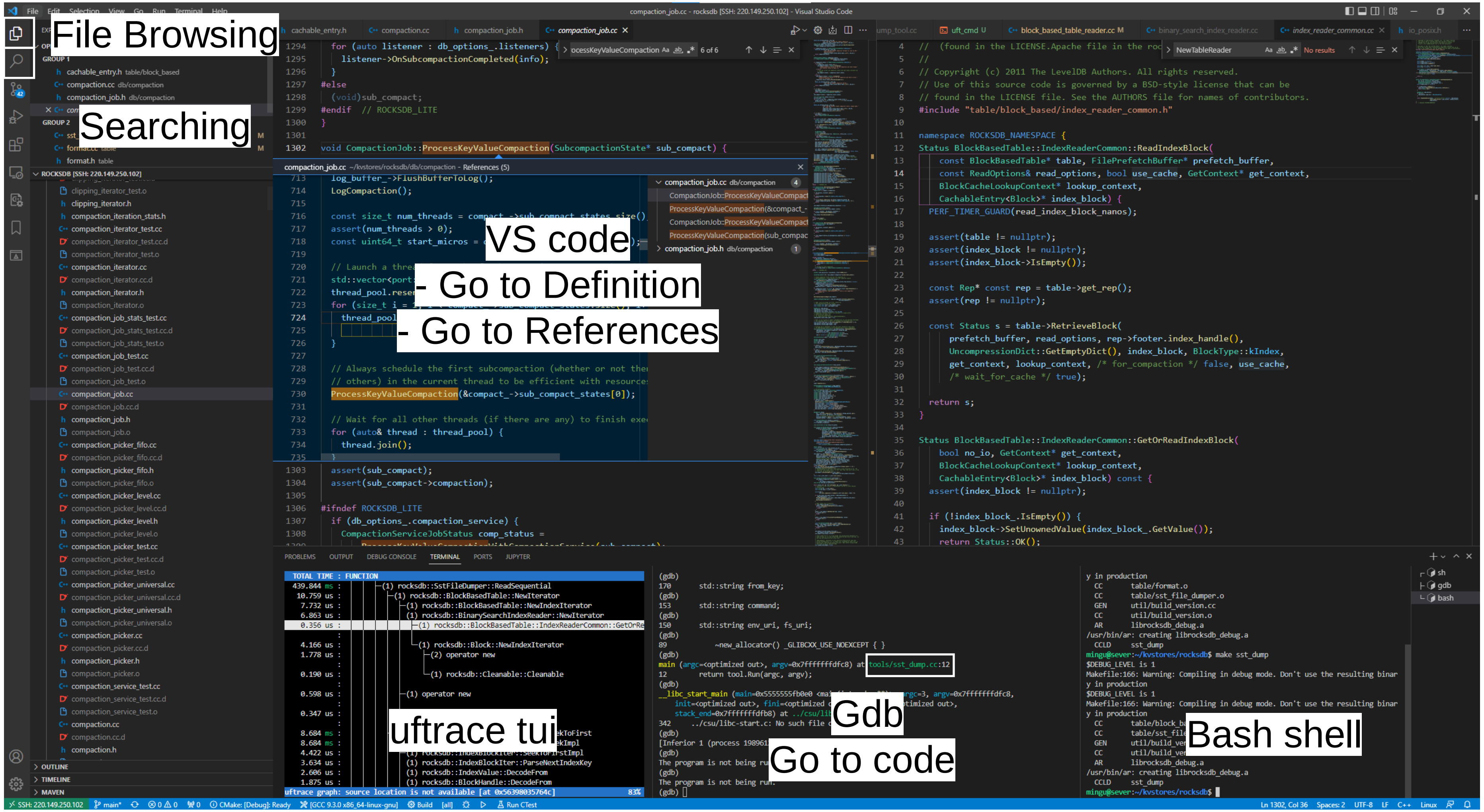
- Tui: SkipList::Insert

```
                ========== Back-trace ==========
  5.188  s : (100000) leveldb::SkipList::Insert
  5.188  s : (100000) leveldb::MemTable::Add
  5.188  s : (100000) leveldb::_GLOBAL__N_1::MemTableInserter::Put
  5.188  s : (100000) leveldb::WriteBatch::Iterate
  5.188  s : (100000) leveldb::WriteBatchInternal::InsertInto
  5.188  s : (100000) leveldb::DBImpl::Write
  5.188  s : (100000) leveldb::Benchmark::DoWrite
  5.188  s : (100000) leveldb::Benchmark::WriteRandom
  5.188  s : (100000) leveldb::Benchmark::ThreadBody
           :
                ========== Call Graph ==========
  5.188  s : (100000) leveldb::SkipList::Insert
  4.824  s :   ├─(100000) leveldb::SkipList::FindGreaterOrEqual
 13.450 ms :   │ ├─(100000) leveldb::SkipList::GetMaxHeight
           :   │ │
473.144 ms :   │ ├─(2495692) leveldb::SkipList::Node::Next
           :   │ │
  4.121  s :   │ └─(2495692) leveldb::SkipList::KeyIsAfterNode
  3.903  s :   │    (2404364) leveldb::MemTable::KeyComparator::operator()
993.576 ms :   │    ├─(4808728) leveldb::GetLengthPrefixedSlice
           :   │    │
  2.544  s :   │    └─(2404364) leveldb::InternalKeyComparator::Compare
492.141 ms :   │      (2404364) leveldb::_GLOBAL__N_1::BytewiseComparatorImpl::Compare
           :   │
 25.035 ms :   ├─(100000) leveldb::SkipList::RandomHeight
 16.364 ms :   │ (133759) leveldb::Random::OneIn
  4.255 ms :   │ (133759) leveldb::Random::Next
           :   │
 11.891 ms :   ├─(100026) leveldb::SkipList::GetMaxHeight
           :   │
 30.790 ms :   ├─(100000) leveldb::SkipList::NewNode
  5.508 ms :   │ ├─(100000) leveldb::Arena::AllocateAligned
  1.455 ms :   │ │ (890) leveldb::Arena::AllocateFallback
  1.384 ms :   │ │ (890) leveldb::Arena::AllocateNewBlock
```

```
Help: (press any key to exit)

 ARROW         Navigation
 PgUp/Dn
 Home/End
 Enter         Fold/unfold graph or Select session
 G             Show (full) call graph
 g             Show call graph for this function
 R             Show uftrace report
 r             Show uftrace report for this function
 s             Sort by the next column in report
 I             Show uftrace info
 S             Change session
 O             Open editor
 c/e           Collapse/Expand direct children graph
 C/E           Collapse/Expand all descendant graph
 n/p           Next/Prev sibling
 u             Move up to parent
 l             Move to the longest executed child
 j/k           Move down/up
 z             Set current line to the center of screen
 /             Search
 </>/N/P       Search next/prev
 v             Show debug message
 f             Customize fields in graph or report mode
 h/?           Show this help
 q             Quit
```

6.3.4. VS Code with gdb & uftrace



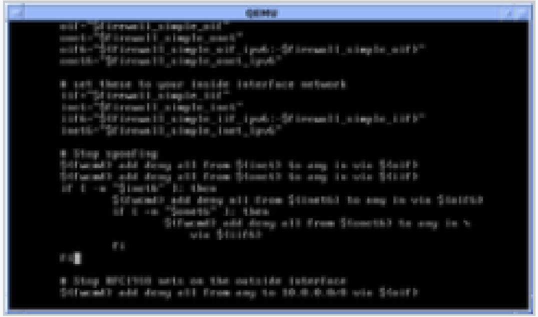
6.4.1. Experimentation and Graphing : shell script

Shell script

19 languages

Article Talk | Read Edit View history Tools

A **shell script** is a computer program designed to be run by a Unix shell, a command-line interpreter.[1] The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text. A script which sets up the environment, runs the program, and does any necessary cleanup or logging, is called a **wrapper**.

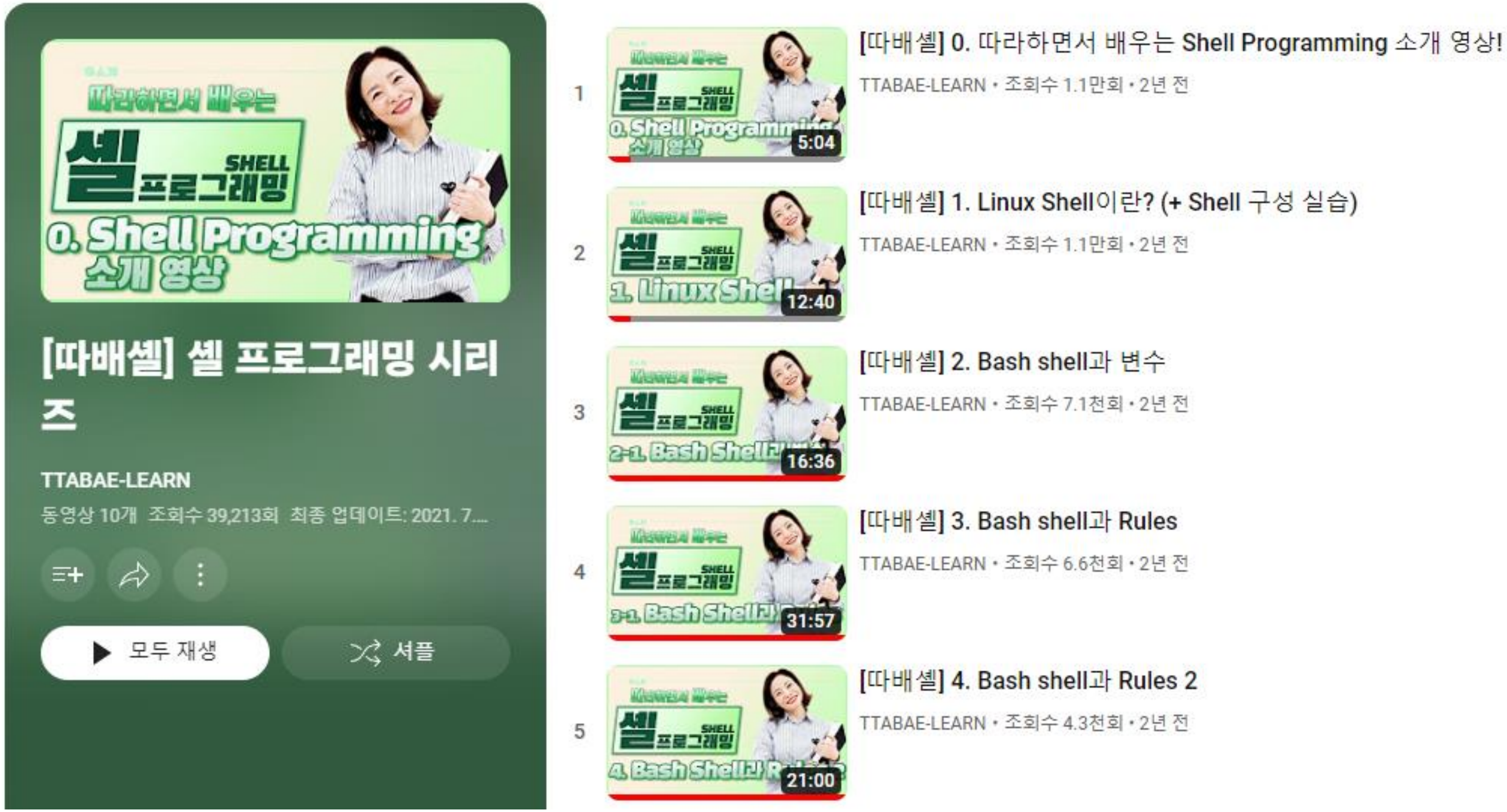


Editing a FreeBSD shell script for configuring ipfirewall

The term is also used more generally to mean the automated mode of running an operating system shell; each operating system uses a particular name for these functions including batch files (MSDos-Win95 stream, OS/2), command procedures (VMS), and shell scripts (Windows NT stream and third-party derivatives like 4NT—article is at cmd.exe), and mainframe operating systems are associated with a number of terms.

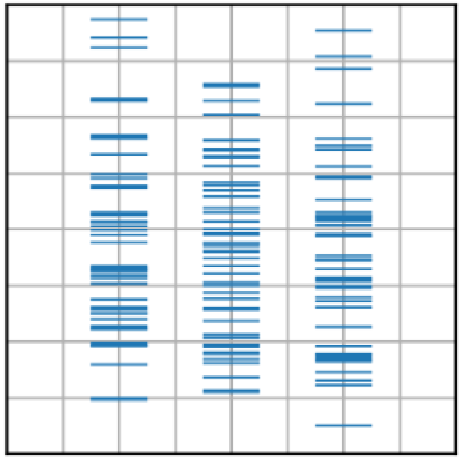
Shells commonly present in Unix and Unix-like systems include the Korn shell, the Bourne shell, and GNU Bash. While a Unix operating system may have a different default shell, such as Zsh on macOS, these shells are typically present for backwards compatibility.

6.4.1. Experimentation and Graphing : shell script



6.4.2. Experimentation and Graphing : *matplotlib*

matplotlib Plot types User guide Tutorials Examples Reference Contribute Releases



eventplot(D)

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Try Matplotlib (on Binder) →

Getting Started | Examples | Reference | Cheat Sheets | Documentation

6.4.2. Experimentation and Graphing : *matplotlib*

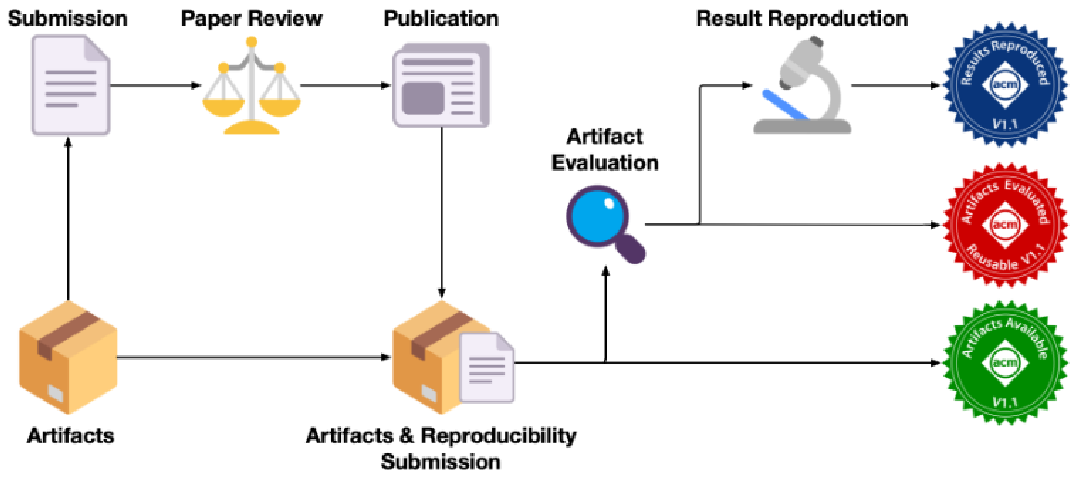
You

Why should I use matplotlib and not excel?

ChatGPT

Using matplotlib instead of Excel for data visualization can be advantageous for several reasons:

1. **Customization and Flexibility:** Matplotlib offers extensive customization options. You can adjust almost every element of your plot, which is particularly useful for creating complex and specialized visualizations.
2. **Integration with Python**: As part of the Python ecosystem, matplotlib integrates seamlessly with other libraries and tools. This is particularly useful for data analysis, allowing you to leverage Python's powerful data manipulation libraries like Pandas and NumPy.
3. **Reproducibility and Automation**: In matplotlib, visualizations are created programmatically, which makes it easy to reproduce and automate them. This is crucial for data analysis tasks that need to be repeated regularly with new data.



6.4.2. Experimentation and Graphing : *matplotlib*

You

Why should I use matplotlib and not excel?

ChatGPT

Using matplotlib instead of Excel for data visualization can be advantageous for several reasons:

4. **Support for Different Plot Types**: Matplotlib supports a wide range of plot types beyond what Excel offers, including 3D plots, interactive plots, and complex statistical visualizations.
5. **Large Community and Resources**: Being a popular open-source tool, matplotlib has a large community of users and developers. This means there's a wealth of tutorials, guides, and forums available for troubleshooting and learning.
6. **Cost and Accessibility**: Matplotlib is free and open-source, unlike Excel which requires a paid license. This makes it more accessible, especially for larger teams or organizations.
7. **Handling of Large Datasets**: Matplotlib is generally more efficient than Excel at handling large datasets. Excel can become slow or unresponsive with very large or complex datasets.

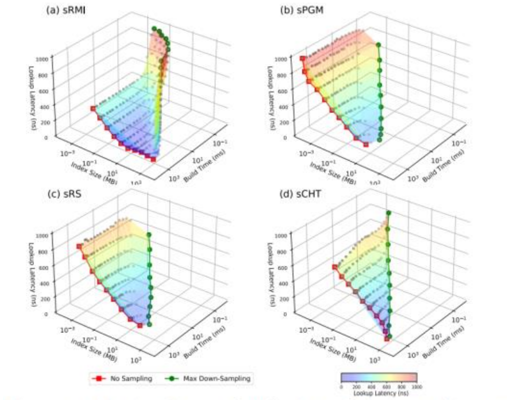


Figure 11: Design Space of indexing schemes on history dataset.

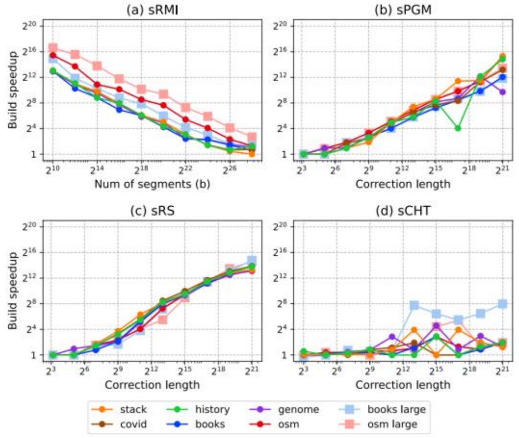


Figure 10: Build speedup with the safe interval.

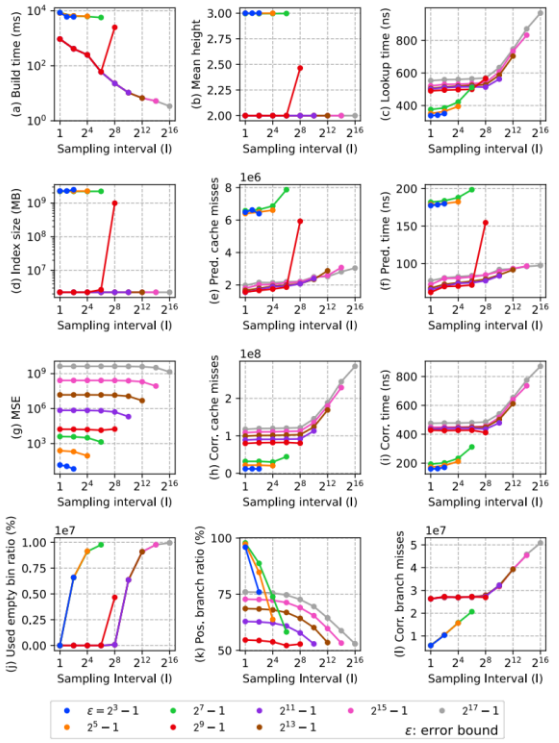
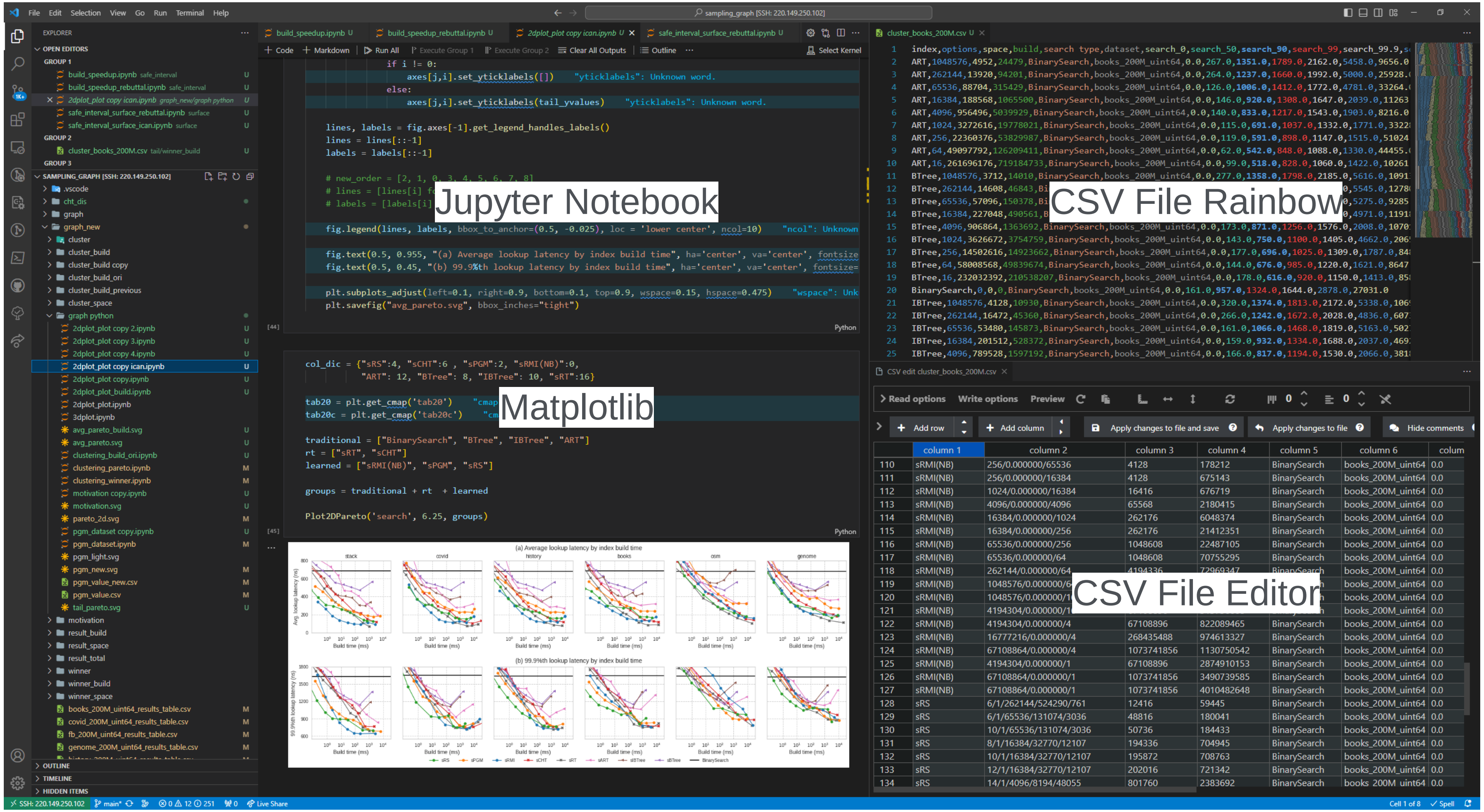


Figure 9: Performance impact of sampling on sCHT on history dataset.

6.4.2. VS Code with matplotlib & csv



6.5. How to Read Papers

How to Read a Paper

S. Keshav
David R. Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, Canada
keshav@uwaterloo.ca

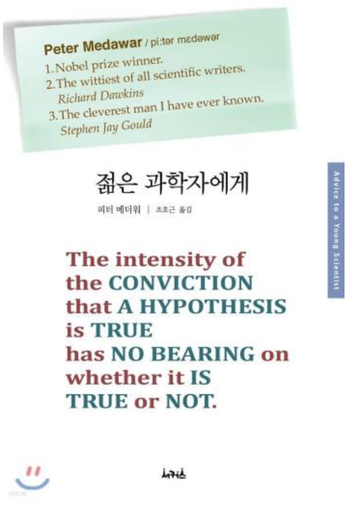
- S. Keshav, "How to Read a Paper“
  - Researchers must read papers for several reasons
    - to review them for a conference or a class
    - to keep current in their field
    - a literature survey of a new field.
    - spend hundreds of hours every year reading papers.
  - Learning to efficiently read a paper is a critical but rarely taught skill.
    - Students waste much effort in the process and are frequently driven to frustration.
  - THE THREE-PASS APPROACH
    - The first pass : a general idea about the paper.
    - The second pass : grasp the paper’s content, but not its details.
    - The third pass : helps you understand the paper in depth.

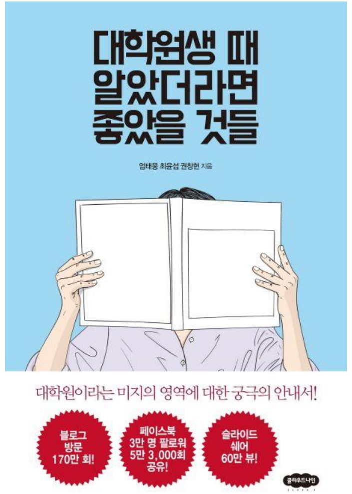
Srinivasan Keshav

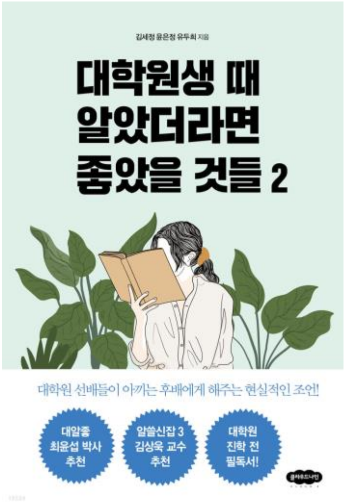
Professor of Computer Science, University of Cambridge
Verified email at cam.ac.uk - Homepage
Computer networking Energy systems

I am the Robert Sansom Professor of Computer Science in the Department of Computer Science and Technology at the University of Cambridge and a Fellow of Fitzwilliam College. I am a Fellow of the Royal Society of Canada, ACM, and IEEE.

6.6. Mindset







박사과정 학생이 유의해야 하는 점

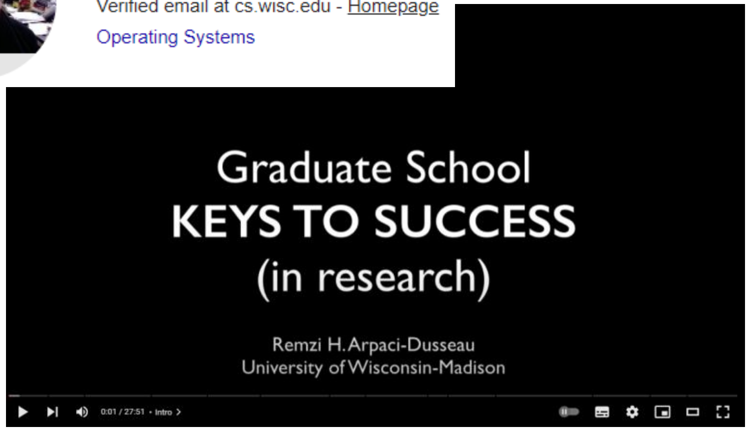
BY 권창현 · PUBLISHED DECEMBER 9, 2011 · UPDATED OCTOBER 26, 2015

"나는 열심히 하는 데, 지도교수는 자꾸 이상한 소리만 하고, 교수가 졸업 준비를 시켜주지 않는다."

이 글을 읽는 분 중에 이와 같은 생각을 해 본 적이 있는 박사과정 학생이 있다면, 아마 이 글이 도움될지도 모르겠다. 조금 더 자세하게는 **"나는 교수가 하라는 대로 이것도 하고 저것도 하고, 정말 열심히 했는데, 교수가 자꾸 논문 방향을 이리저리 바꾸기만 하고, 논문 진도는 안 나가고, 도대체 교수는 생각이 있는 건지, 이 교수 밑에서 배울 게 있는 건지, 내가 졸업이나 할 수 있을런지도 모르겠다"**라고 생각한 적이 있다면, 이 글이 확실히 도움이 될 것 같다.

Changhyun Kwon
Department of Industrial and Systems Engineering, KAIST
Verified email at kaist.ac.kr - Homepage
Transportation Operations Research Game Theory Risk Management

Remzi Arpaci-Dusseau
University of Wisconsin-Madison
Verified email at cs.wisc.edu - Homepage
Operating Systems



논문 제 1 저자의 책임

Professional · 2016. 8. 20. 06:11

제 1 저자는 모든 과정을 주도적으로 책임져야 한다. 아이디어를 내고, 실험/증명을 하고, 논문을 썼다고 해도 리뷰를 받고 나서의 후속 작업도 만만치 않다. 논문이 되면 다행이지만, 떨어지면 수정보완해서 다른 곳에 제출해야한다. 한두달 사이에 끝날 일이 아니다.

Sue Moon
School of Computing, KAIST
Verified email at kaist.edu - Homepage
Online social networks networking systems

Welcome 🎉 & Work Hard 😎

Thank you!