

# 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?
2. Read-only learned indexes
3. Applying sampling for learned indexes
4. Updatable learned indexes
5. Research Topics
6. Tips & Tools

# **1. What is learned index?**

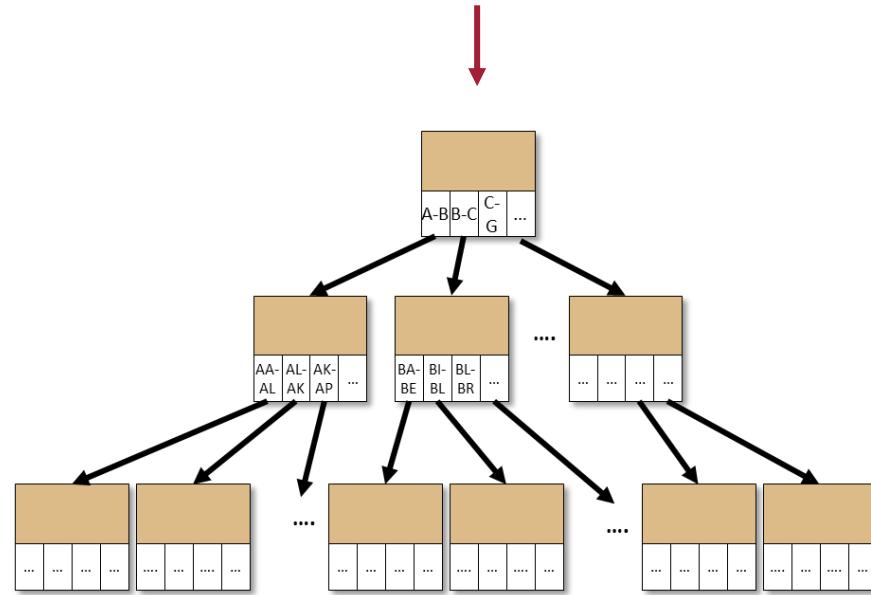
# Exploiting the Data Distribution

<b>id</b>	<b>date</b>	<b>first_name</b>	<b>last_name</b>	<b>email</b>	<b>address</b>	<b>zip</b>	<b>state</b>	<b>credit_card_nb</b>	<b>amount</b>
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'Curriган	docurriган4@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	Boyett	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 Prairiewood 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

## Key

(e.g., id, name, address,...)



## Position

<b>id</b>	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	...	800M
-----------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-----	------

# Traditional B-Tree Index

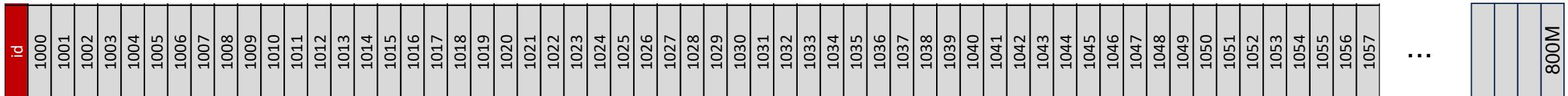
# Key

(e.g., id, name, address,...)



array[id - 1000]

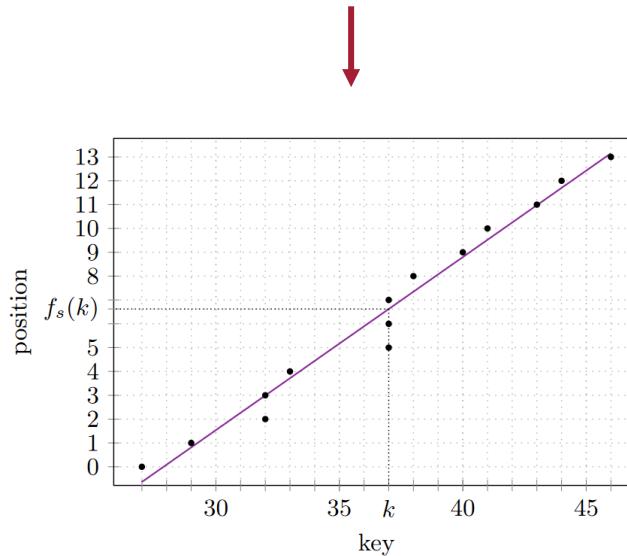
## Position



# Learned Index

# Key

(e.g., id, name, address,...)

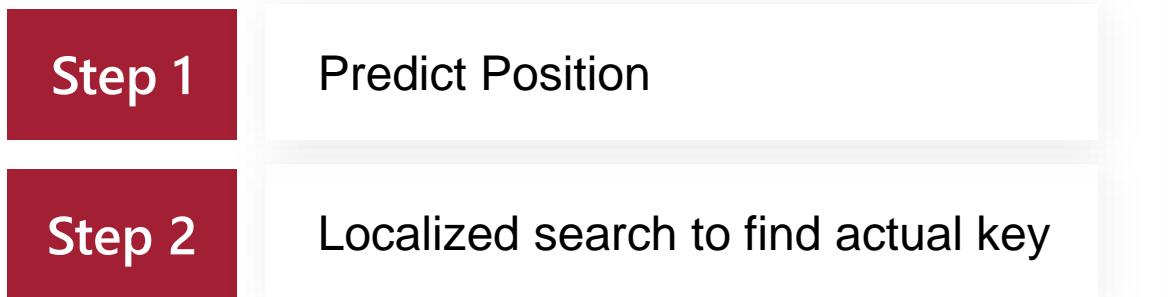


## Position

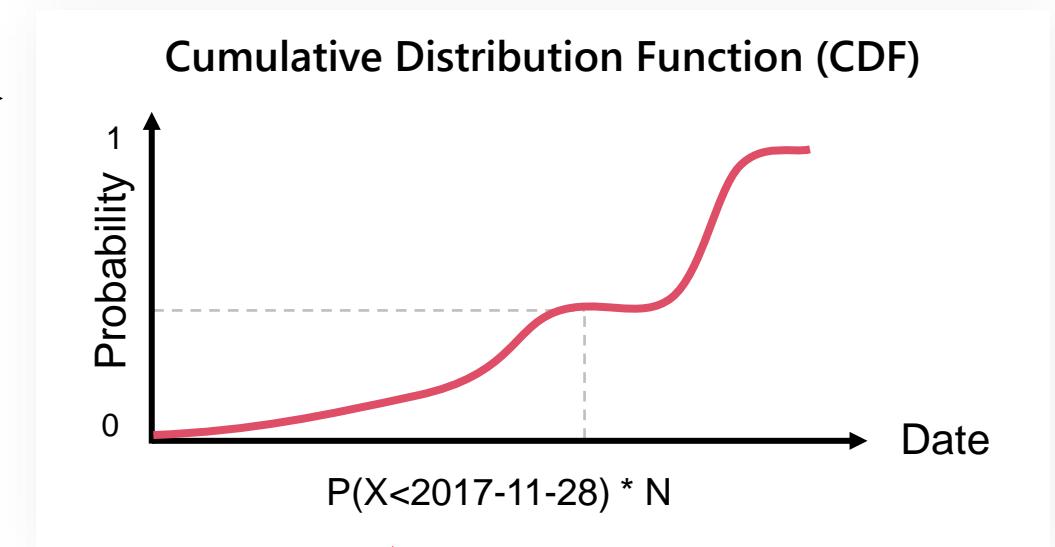
Id	Value
1000	1000
1001	1001
1002	1002
1003	1003
1004	1004
1005	1005
1006	1006
1007	1007
1008	1008
1009	1009
1010	1010
1011	1011
1012	1012
1013	1013
1014	1014
1015	1015
1016	1016
1017	1017
1018	1018
1019	1019
1020	1020
1021	1021
1022	1022
1023	1023
1024	1024
1025	1025
1026	1026
1027	1027
1028	1028
1029	1029
1030	1030
1031	1031
1032	1032
1033	1033
1034	1034
1035	1035
1036	1036
1037	1037
1038	1038
1039	1039
1040	1040
1041	1041
1042	1042
1043	1043
1044	1044
1045	1045
1046	1046
1047	1047
1048	1048
1049	1049
1050	1050
1051	1051
1052	1052
1053	1053
1054	1054
1055	1055
1056	1056
1057	1057

800M

# Learned Index



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



date	2017-01-01	2017-01-02	2017-01-03	2017-01-03	2017-01-04	2017-01-04	2017-01-05	2017-01-05	2017-01-06	2017-01-07	2017-01-09	2017-01-09	2017-01-09	2017-01-10	2017-01-10	2017-01-11	2017-01-11	2017-01-12	2017-01-12	2017-01-13	2017-01-14	2017-01-15	2017-01-16	2017-01-17	2017-01-18	2017-01-19	2017-01-20	2017-01-21	2017-01-22	2017-01-22	2017-01-22	2017-01-23	2017-01-24	2017-01-24	2017-01-26	2017-01-26	2017-01-28	2017-01-29	2017-01-30	2017-01-30	2017-01-30	2017-01-31	2017-01-31	2017-02-01	2017-02-01	2017-02-02	2017-02-04	2017-02-05	2017-02-05	2017-02-06	2017-02-06	2017-02-07	2017-02-07	2017-02-08	2017-02-08	2017-02-08	2017-02-09	...	2017-11-27	2017-11-27	2017-11-27	2017-11-28	2017-11-28
------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	-----	------------	------------	------------	------------	------------

# Exploiting the Data Distribution

<b>id</b>	<b>date</b>	<b>first_name</b>	<b>last_name</b>	<b>email</b>	<b>address</b>	<b>zip</b>	<b>state</b>	<b>credit_card_nb</b>	<b>amount</b>
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	docurriagan4@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	amzn				1 Badeau Place			\$ 854.00
1010	2017-01	face				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	osm				80 Novick Road			\$ 954.00
1016	2017-01	wiki				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			\$ 474.00
1024	Various real-world integer/decimal dataset					204 Huxley Road			\$ 83.00
1025	2017-01-20	Muncy	Higgin	ahiggin@pewtrusts.org.au	Debra Pass	74184	Oklahoma	3538-604-6315-110	\$ 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	Ienks	dienksx@virginia.edu	1 Butterfield Road	85210	Arizona	3578-9195-0297-7730	\$ 636.00

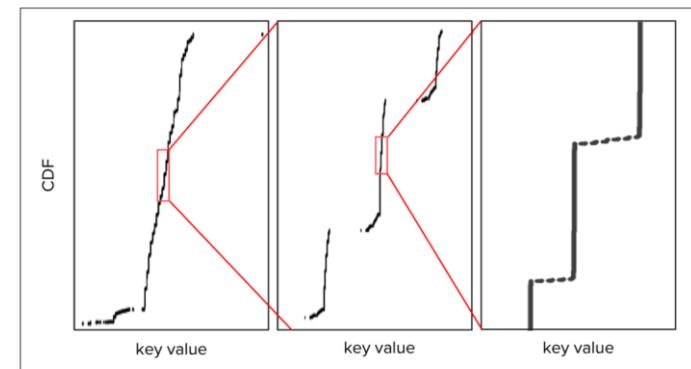
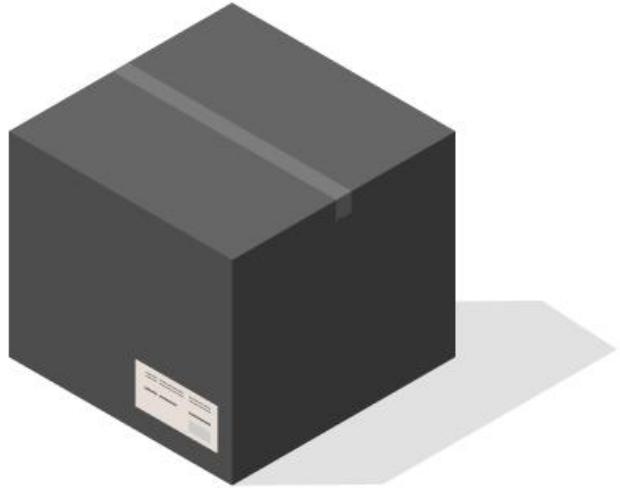


Figure 1-2: Repeated zooming on the CDF of the Wiki dataset reveals its fractally stepwise nature.

Real world string dataset (wiki)

# Key Insight of Learned Index



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

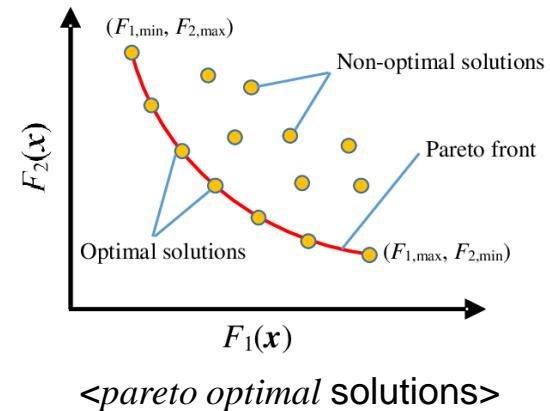
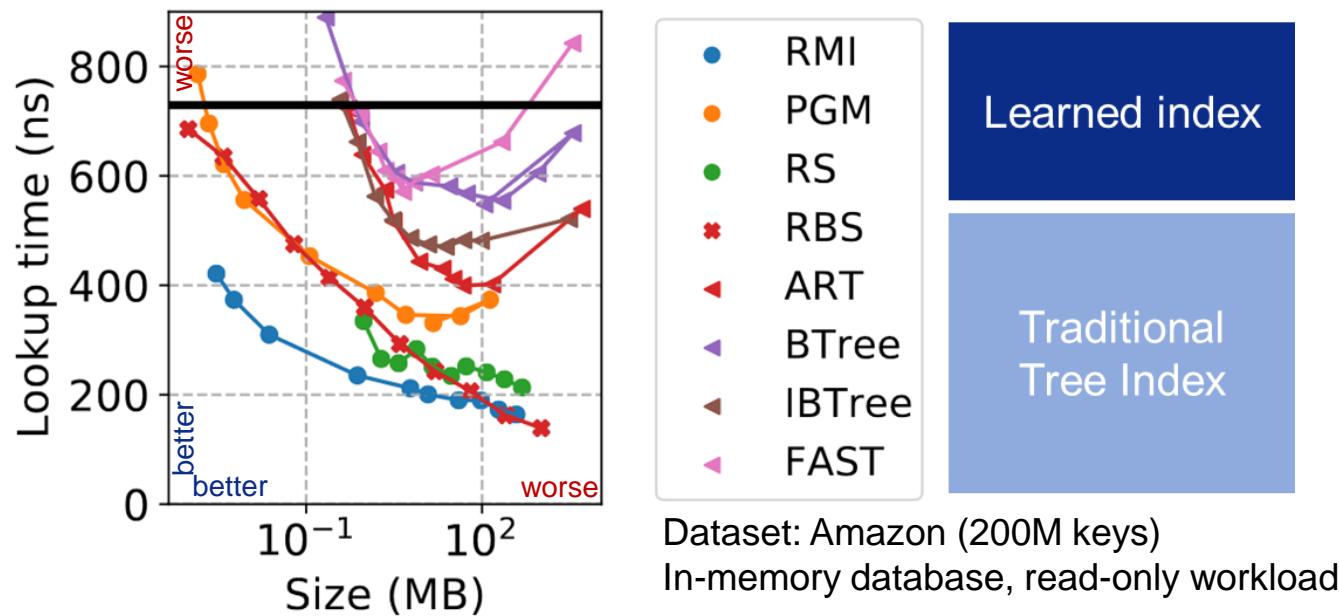


But knowing the data distribution  
might allow for significant  
performance gains and might even  
change the complexity of data structures.

$O(\log n) \rightarrow O(1)$  for lookups  
 $O(n) \rightarrow O(1)$  for storage

## 2.5. Performance

- So, how well does learned index perform?



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)

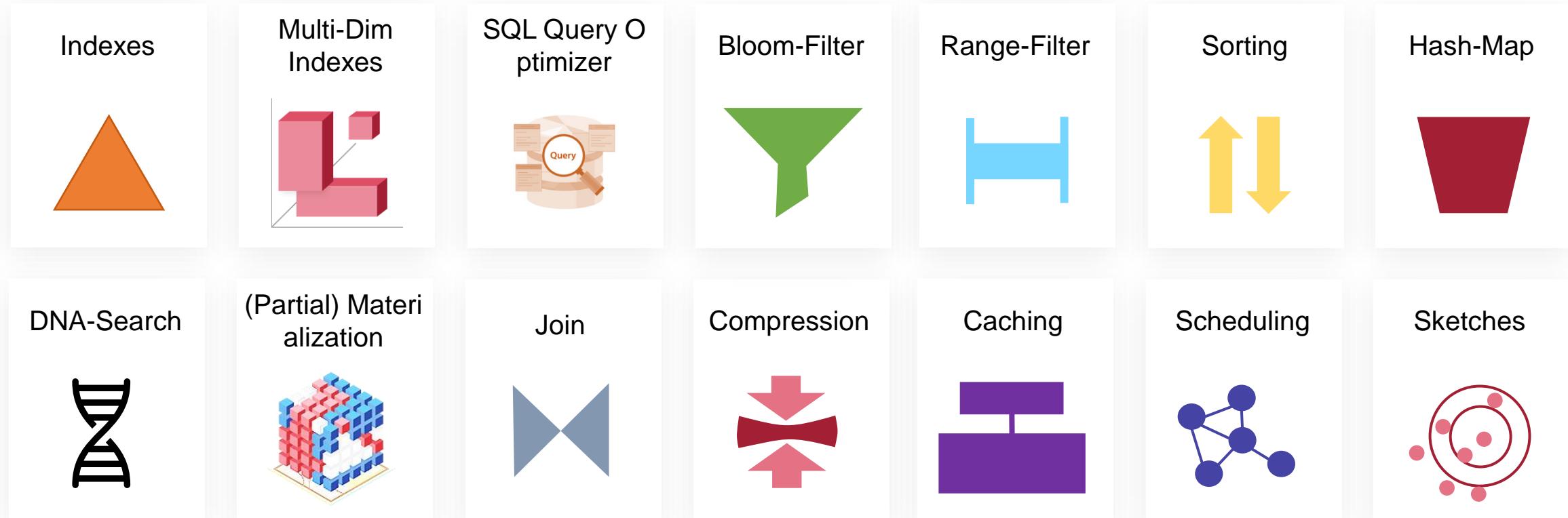
## Index Structure

- 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 Lear
- Spatial Interpolation-based Learned
- APEX: A High-Performance Learned I
- RUSL: Real-time Updatable Spline L
- PLEX: Towards Practical Learned Inde
- SPRIG: A Learned Spatial Index for R
- 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 Index (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 <https://github.com/LumingSun/ML4DB-paper-list>
- PLIN: A Persistent Learned Index for Recovery (VLDB 2023)
- WIPE: a Write-Optimized Learned Index for Persistent Memory (TACO 2023)



# Types of Instance-Optimized components

- Instance-Optimized algorithms, data structures, and components



... and more

1. What is learned index?
2. Read-only learned indexes
3. Applying sampling for learned indexes
4. Updatable learned indexes
5. Research Topics
6. Tips & Tools

## **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 :  $\forall k \in D, \text{Error}(k) \leq \varepsilon$
  - $k$  exist in correction range ( $= [p - \varepsilon, p + \varepsilon]$ )  $\rightarrow$  binary search

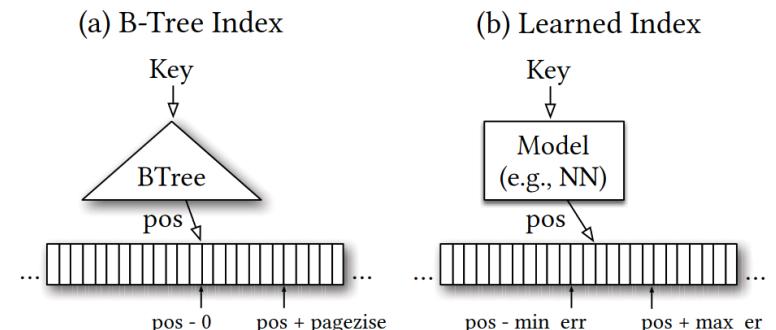
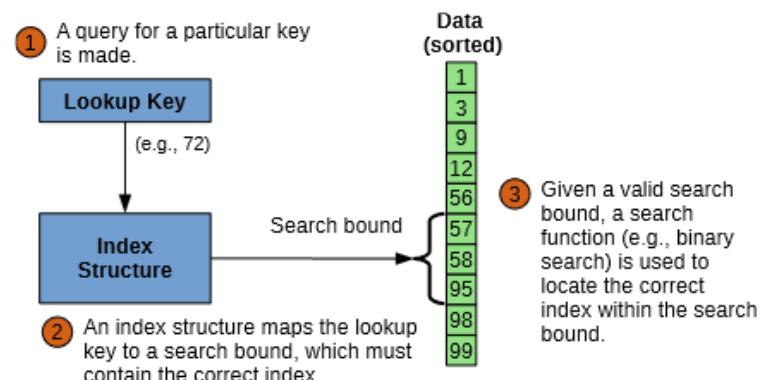


Figure 1: Why B-Trees are models



# 2.1. Index Definition

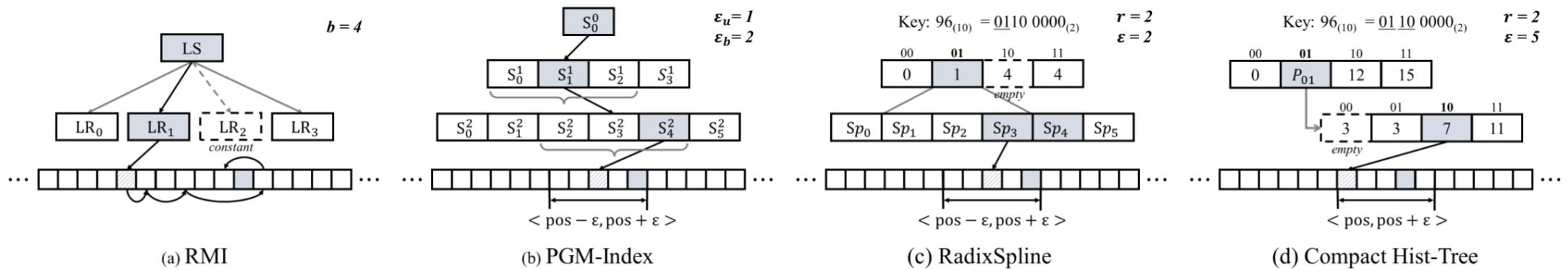


Figure 1: Structure of indexes: RMI, PGM, RS, and CHT.

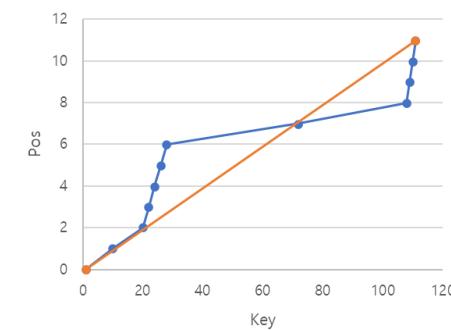
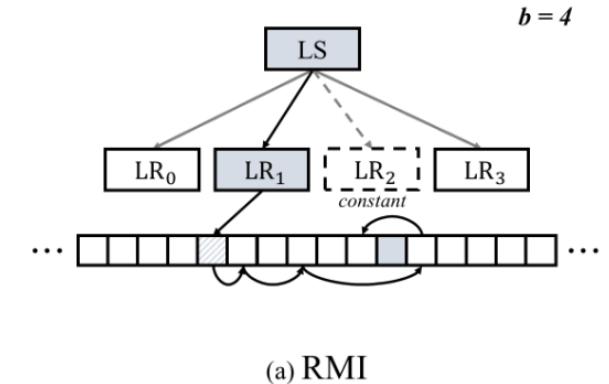
Index	Model				Building			Correction		
	Top (Upper) Layer	Bottom Layer	# of Layers	Param.	Training Algorithm	Segmentation	Order	Error Bounded	Search Algorithm	Correction Range
RMI [13]	Linear Spline	PLR	2	b	Linear Least Squares	fixed	↓	X	Exponential	-
PGM [8]	EB-PLR	EB-PLR	$\geq 2$	$\varepsilon_u, \varepsilon_b$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

Table 1: Summary of the Indexes.

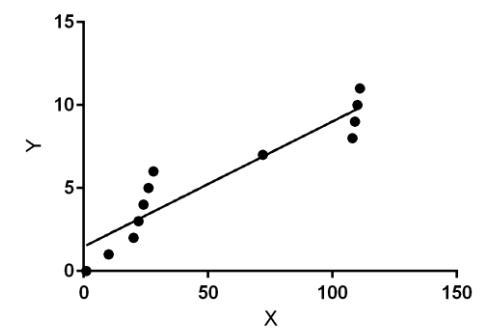
# 2.2. RMI (Recursive Model Index)

## ■ Structure

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

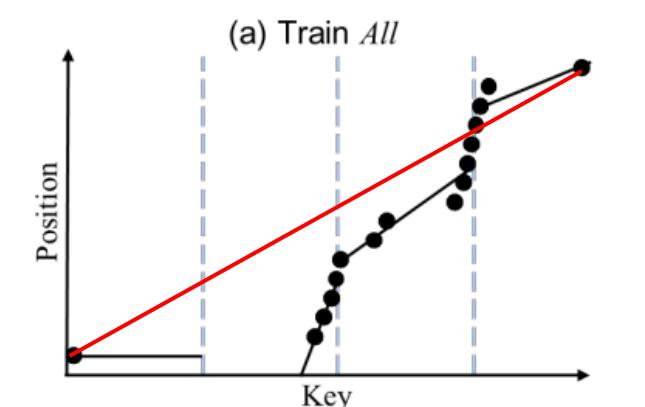
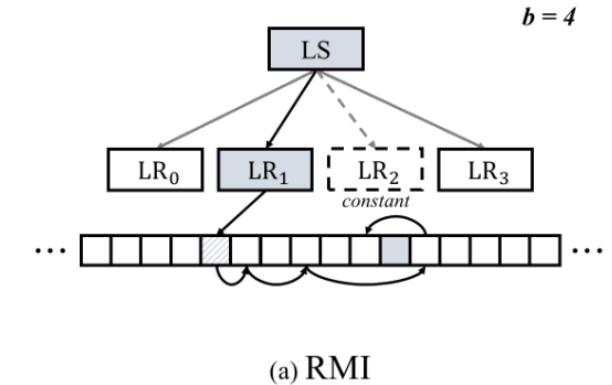


Linear Regression

# 2.2. RMI (Recursive Model Index)

## ■ 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

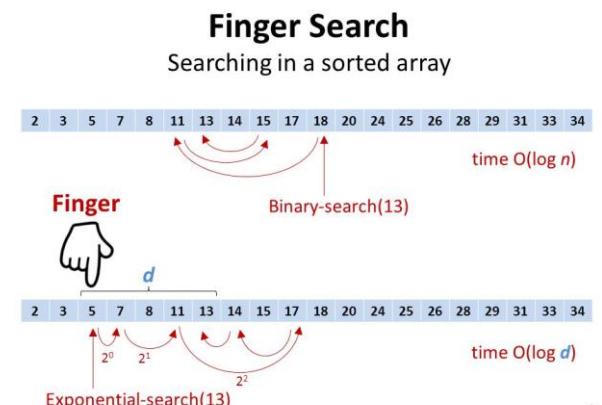
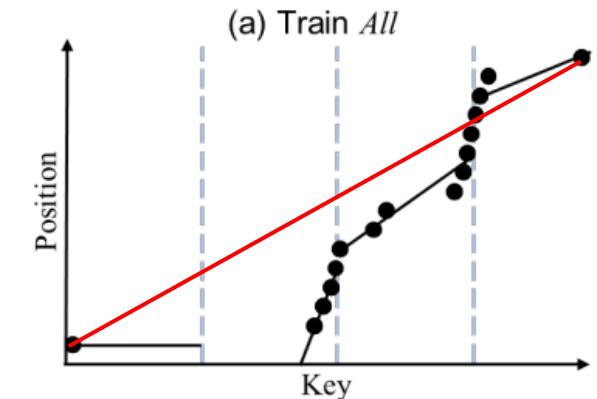
- Prediction
  - Using LS model in top layer,
    - selects a bottom segment
  - Using chosen linear segment in bottom layer,
    - predicts the final position of the key in the dataset.
- Correction
  - If error bound is available
    - binary search on correction range (error-bound).
  - If not
  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

$$\llbracket p \rrbracket_a^b := \max(a, \min(p, b)). \quad (2)$$

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

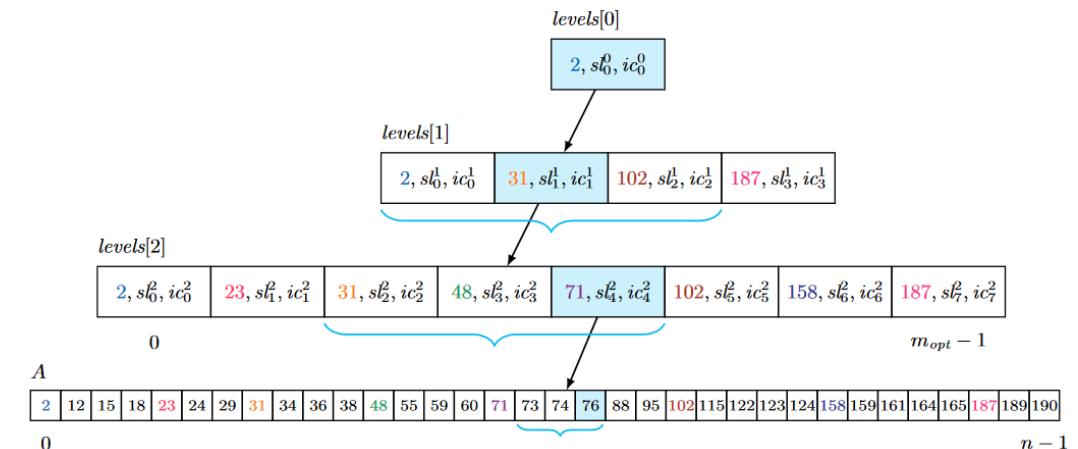
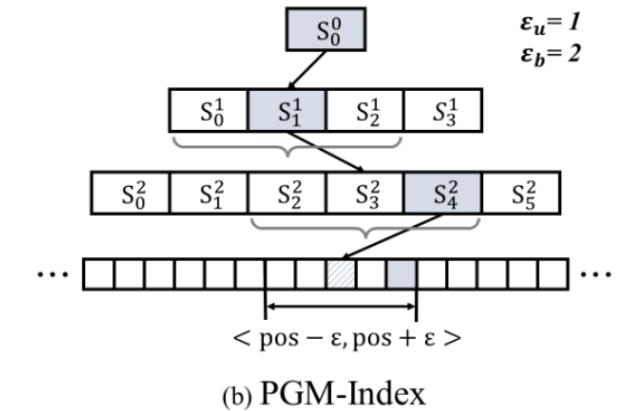
$$f_i(x) = \begin{cases} f_0^0(x) & i = 0 \\ f_i^{\lfloor \lfloor l_i \rfloor \cdot f_{i-1}(x)/n \rfloor^{l_i-1}}(x) & 0 < i < k \end{cases} \quad (3)$$



# 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 ( $\varepsilon$ ) in all predictions
  - the bottom layer's error bound :  $\varepsilon_{bottom}$
  - the upper layers' error bound :  $\varepsilon_{upper}$



## 2.3. 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

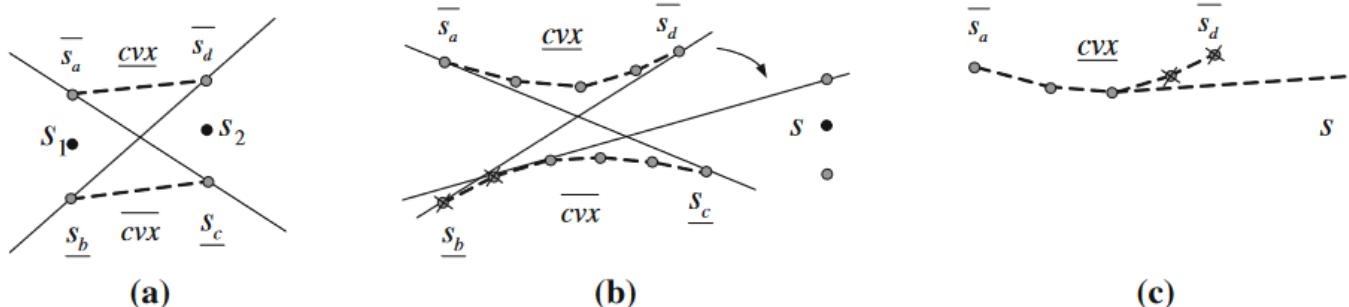
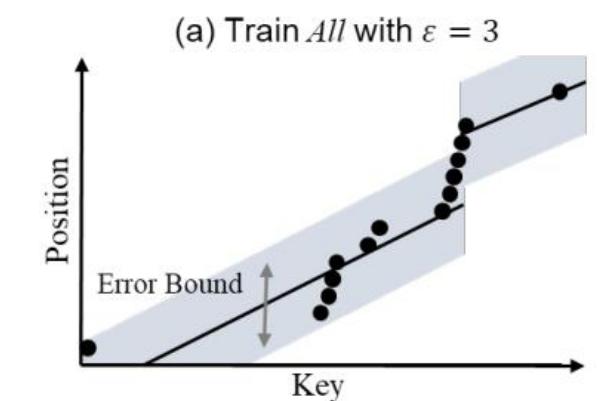
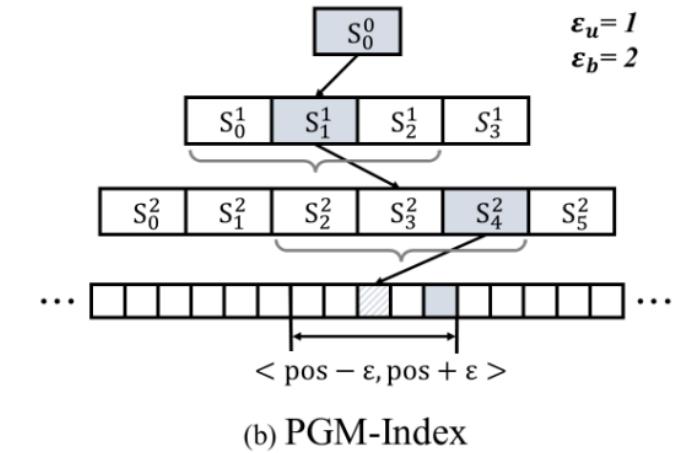


Fig. 7 Optimal algorithm: a Initialization; b Updating of extreme slopes; c Updating of convex hull

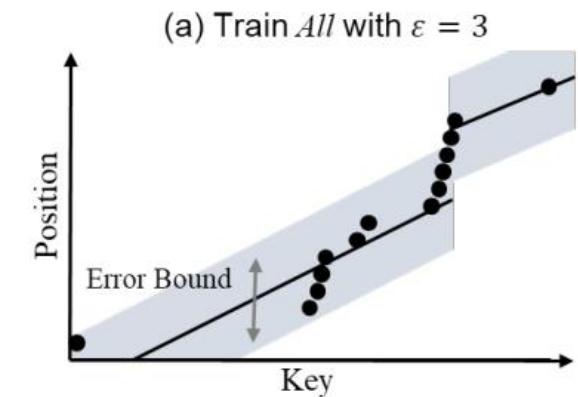
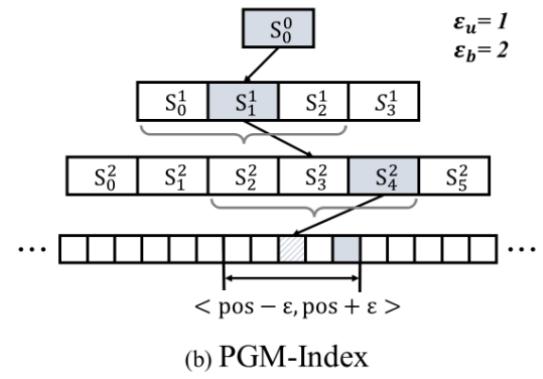


Linear Regression Segments in Bottom Layer of PGM

## 2.3. PGM-Index

- **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.

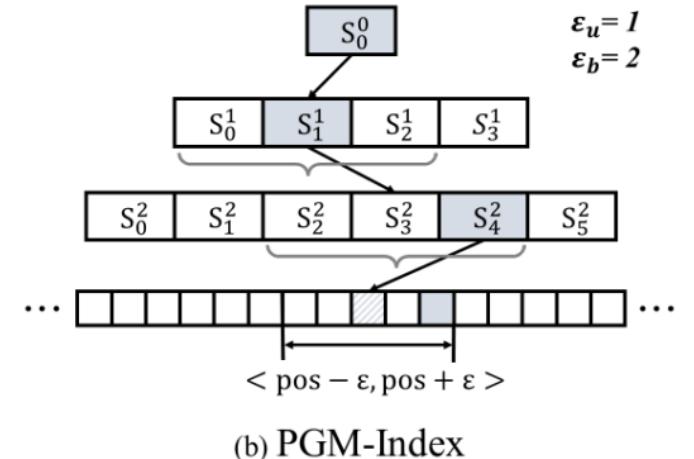


Linear Regression Segments in Bottom Layer of PGM

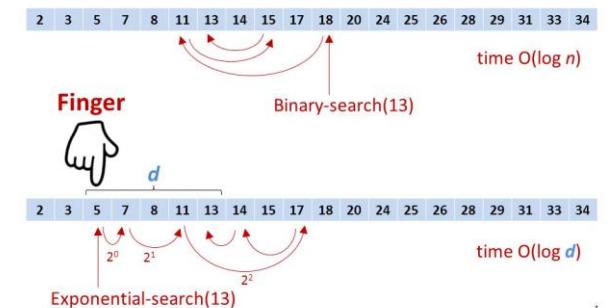
## 2.3. PGM-Index

### ■ 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



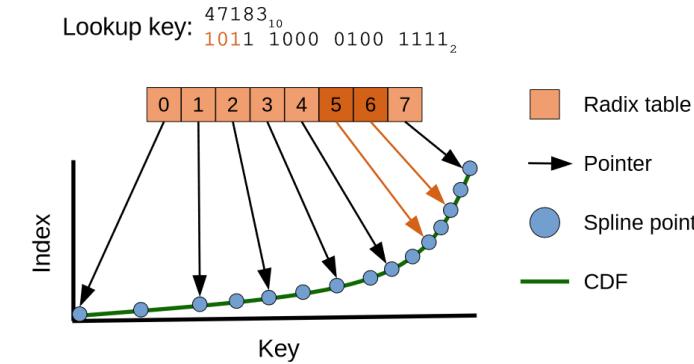
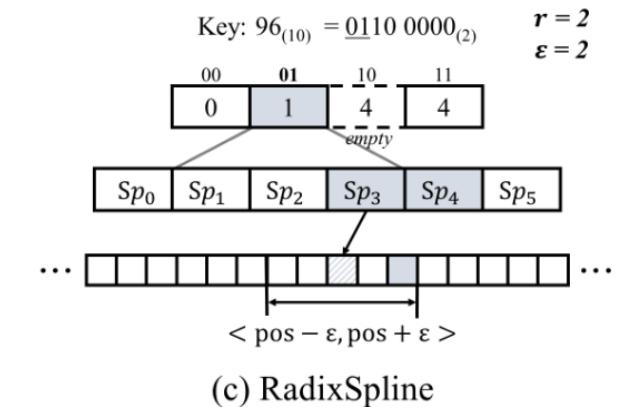
**Finger Search**  
Searching in a sorted array



# 2.4. RS (RadixSpline)

## ■ Structure

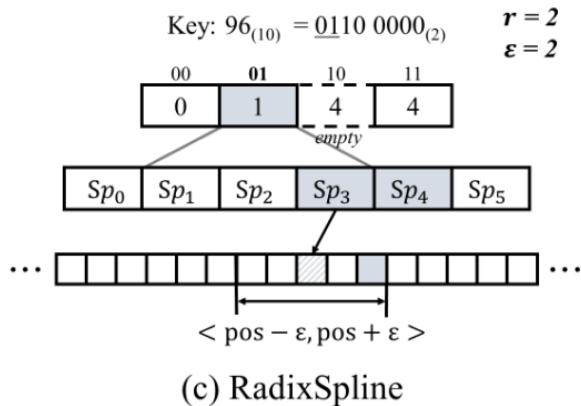
- a two-layered learned index
  - The bottom layer : EB-PLS.
    - error-bounded piecewise linear splines
    - guarantees an error-bound( $\varepsilon$ ) for the prediction.
  - The top layer : radix table
    - A histogram node indexes bottom linear splines
    - divides the key range into a fixed number of  $2^r$  bins with equal width
      - ✓  $r$  : (prefix) radix bits



# 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



```

GreedySplineCorridor
Input: a spline  $S$ ,  $|S| = n$  and an error corridor size  $\epsilon$ 
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  $\overline{BL}$ 
     $B = S[i-1], R = Ro < B >$ 
     $U = C + \epsilon, L = C - \epsilon$ 
  else
     $U' = C + \epsilon, L' = C - \epsilon$ 
    if  $\overline{BU}$  is left of  $\overline{B'U'}$ 
       $U = U'$ 
    if  $\overline{BL}$  is right of  $\overline{B'L'}$ 
       $L = L'$ 
   $R = Ro < S[n] >$ 
return  $R$ 

```

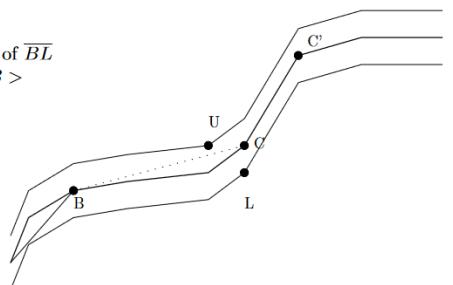


Fig. 1. Greedy Spline Approximation with a Given Error Corridor

# 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

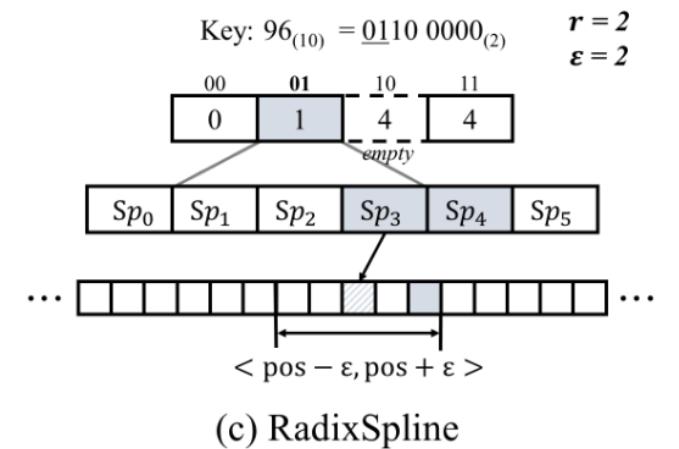
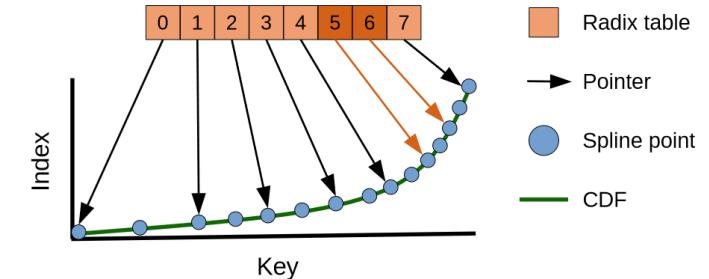
- EB-PLS (Bottom)

- pinpointing two contiguous splines surrounding the lookup key.
- the key's position is predicted through spline interpolation.

- Correction

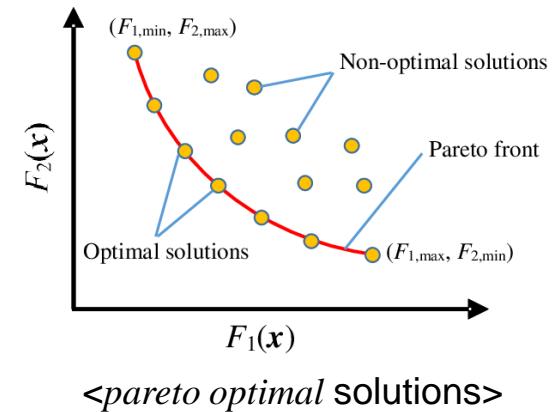
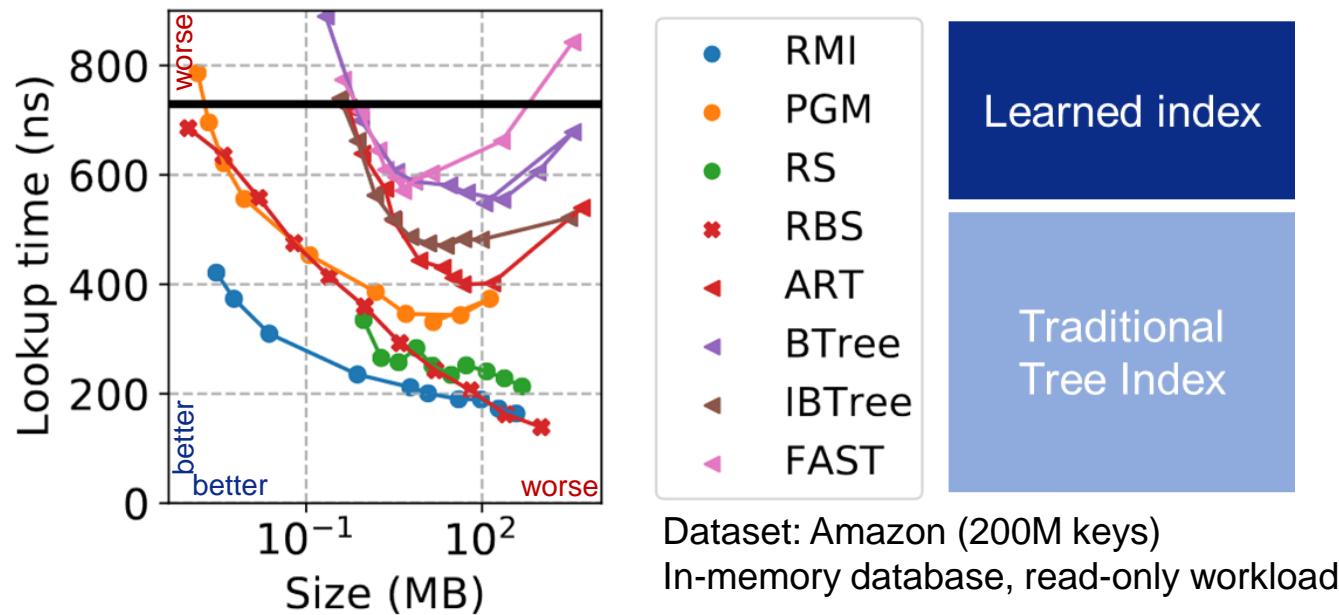
- Binary search the key only within the correction range
  - EB-PLS also provides an error-bound for prediction

Lookup key:  $47183_{10}$   
 $1011\ 1000\ 0100\ 1111_2$



## 2.5. Performance

- So, how well does learned index perform?

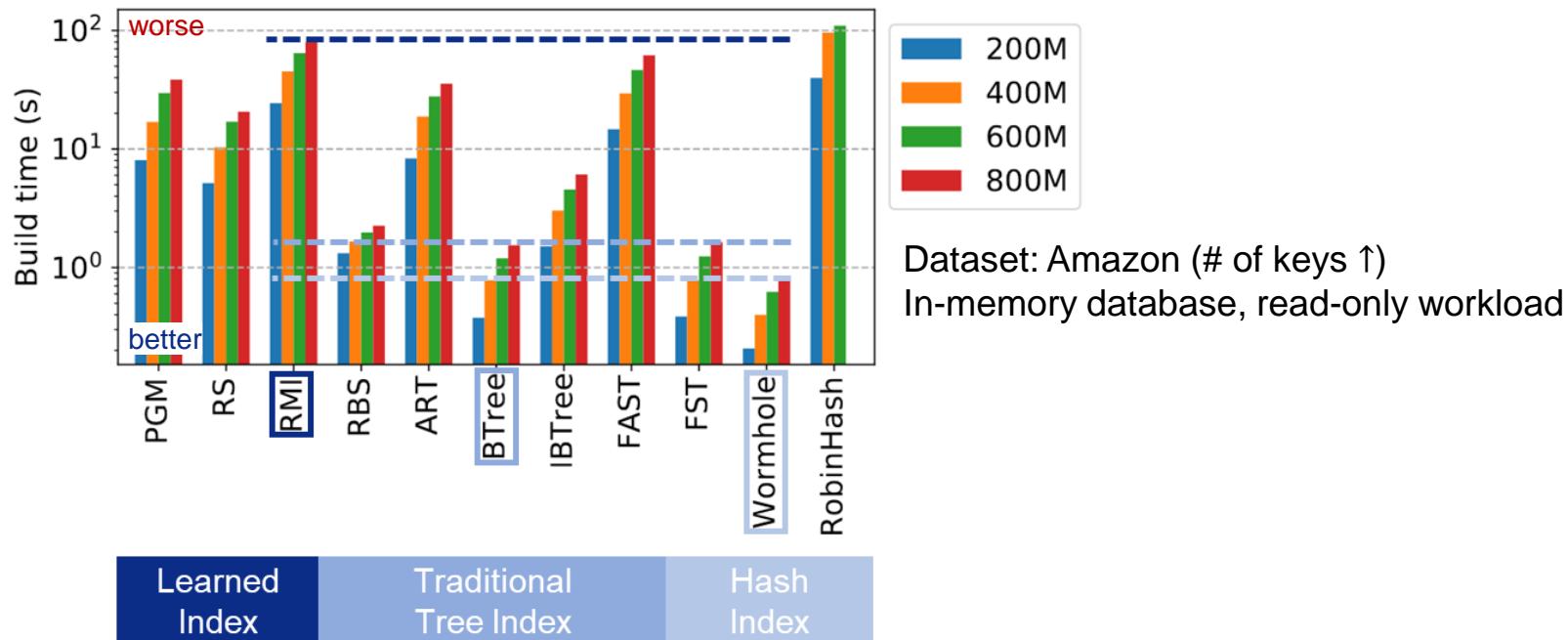


Learned indexes are *pareto optimal (space-efficient)* for size and lookup.

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

But, learned index is not practical yet.



**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

1. What is learned index?
2. Read-only learned indexes
3. Applying sampling for learned indexes
4. Updatable learned indexes
5. Research Topics
6. Tips & Tools

### **3. Applying sampling for learned indexes**



[SIGMOD' 24]

# Can Learned Index be Build-Efficient? A Deep Dive into Sampling Trade-Offs



Minguk Choi<sup>†</sup>



Seehwan Yoo<sup>‡</sup>



Jongmoo Choi<sup>†</sup>

School of Software Science<sup>†</sup>, School of Mobile System Engineering<sup>‡</sup>  
Dankook University

4<sup>th</sup> Round 1<sup>st</sup> Review

Accept through shepherding:

1.4% (4/287)

Major Revision:

28.2% (81/287)

Reject & withdraw:

70.3% (202/287)

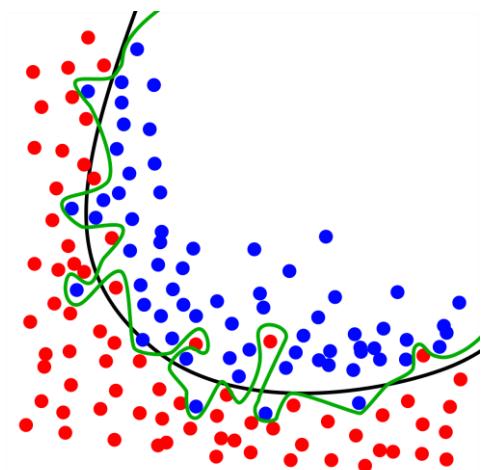
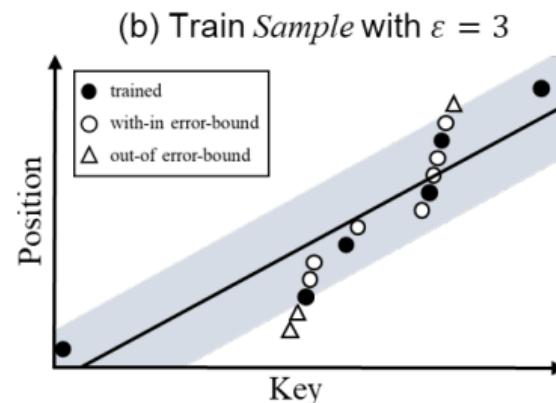
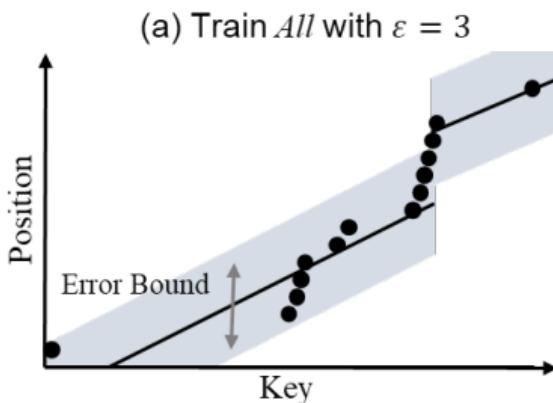


# 3.1. Applying Sampling on Learned Index

- 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

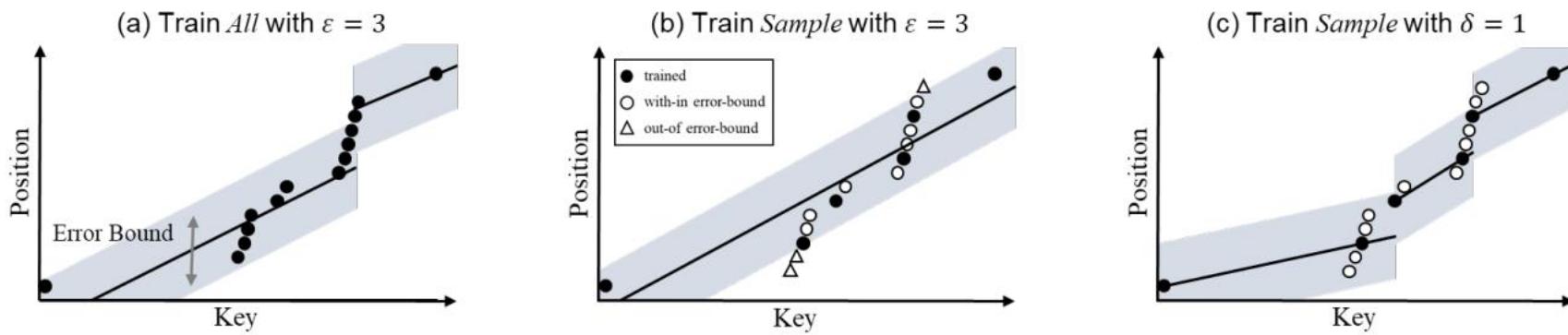
- Error-Bound Property
  - For  $\forall k \in D$ ,  $Error(k) = |Pred(k) - Pos(k)| \leq \varepsilon$
  - important for robustness, especially where correction is expensive. (e.g., disk/remote I/O)
- EB-PLA (Error-Bounded Piece-wise Linear Approximation) Model
  - Learn all the data, dynamically form segments, ensure all errors are less than  $\varepsilon$ .
  - 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  $\forall s \in S$ ,  $Error(s) \leq \delta \rightarrow \text{For } \forall k \in D, \text{Error}(k) \leq \varepsilon (= \delta + I - 1)$
- 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  $Pos(s_{hi})$ . 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  $\delta$ .

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

Through this, the prior bound of  $Pred(k)$  in relation to  $Pos(s_{lo})$ ,  $Pos(s_{hi})$  and  $\delta$  can be transformed into a relation to  $Pos(k)$ ,  $\delta$  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  $\varepsilon$ , and so is  $Error(s)$ . Finally, we get

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

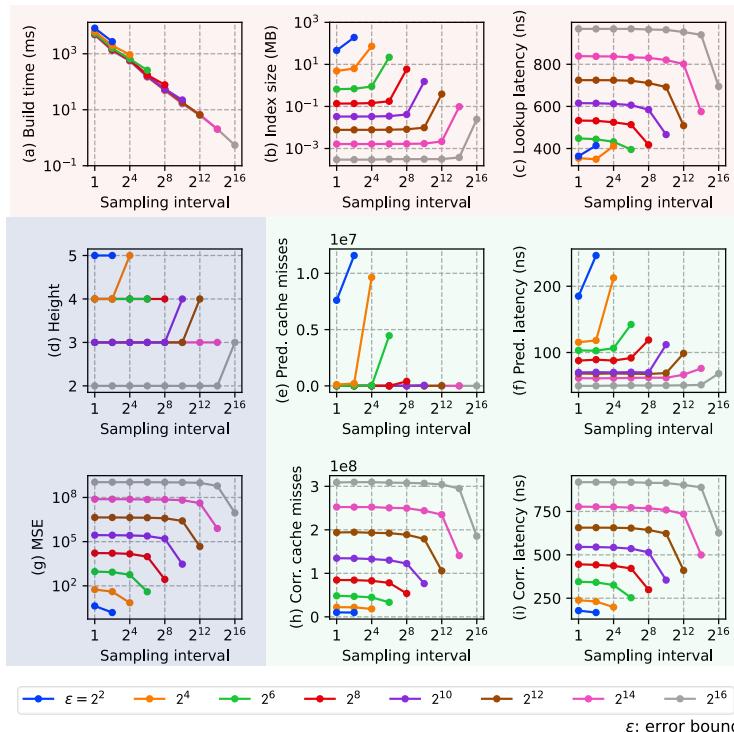
□

# 3.3. Sampling Trade-offs

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

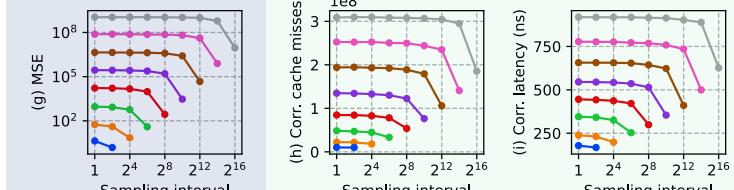
## 1. Index performance

- (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	<b>0.990</b>
Corr. Lat.	0.986	0.993	0.922	<b>0.995</b>

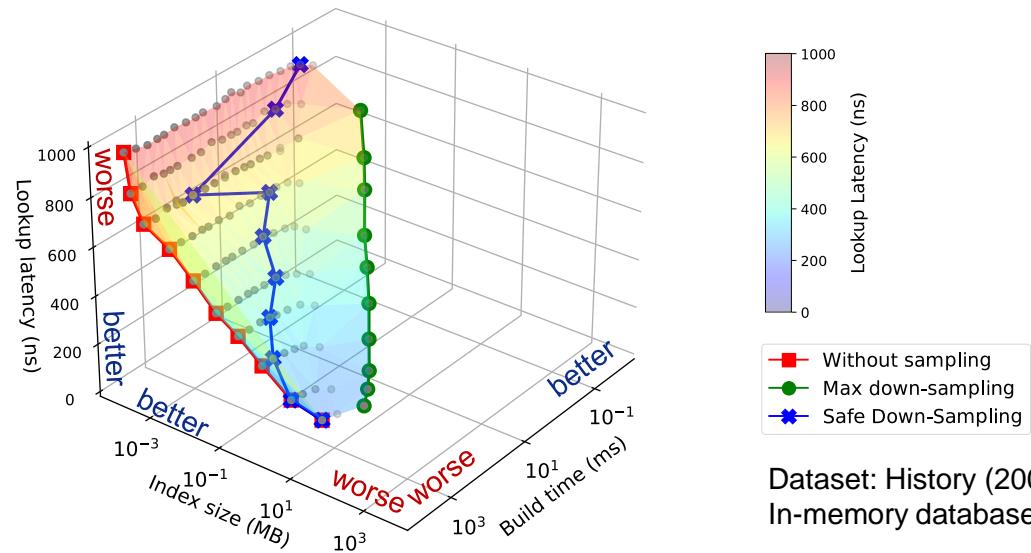
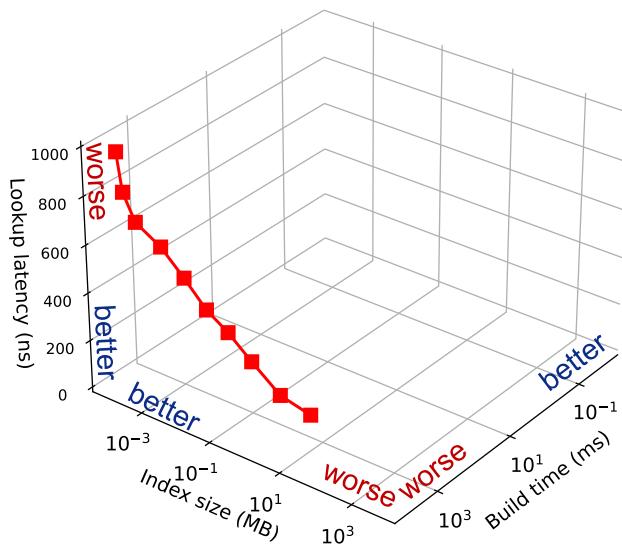
Table 2: Correlation coefficient between the performance counters with lookup latency.

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  $\mu$ s).  
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.**

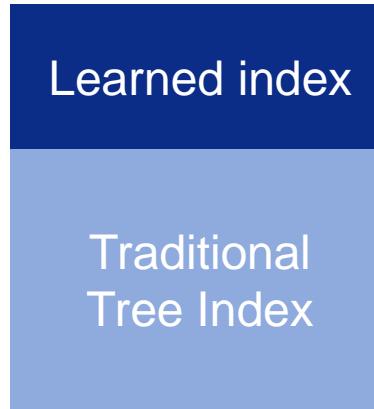
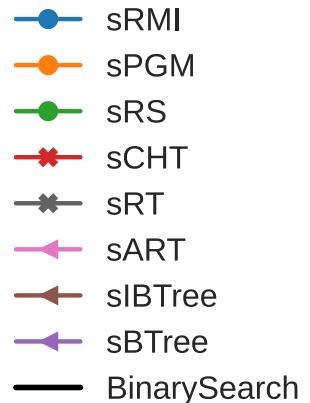
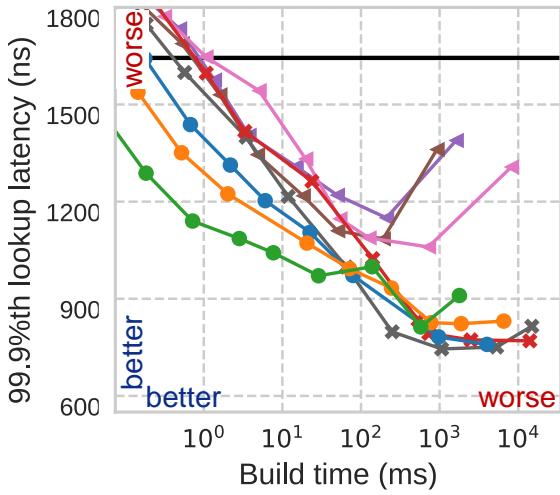
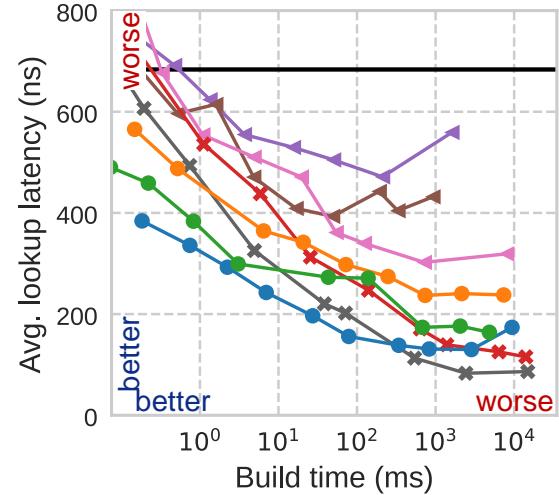


Dataset: History (200M keys)  
In-memory database, read-only workload

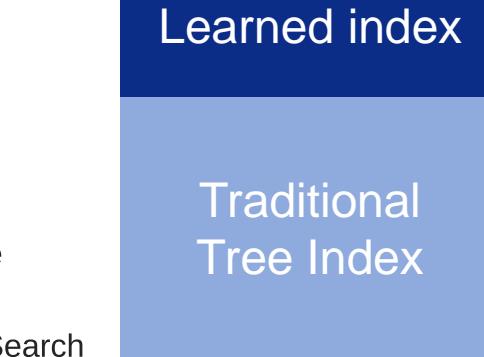
Sampling introduces a trade-off between build/size/lookup → **broadening the design space.**

**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>



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**.

1. What is learned index?
2. Read-only learned indexes
3. Applying sampling for learned indexes
4. Updatable learned indexes
5. Research Topics
6. Tips & Tools

## **4. Updatable learned indexes**

# Various Learned Indexes

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

Learned	Index	Insert		Lookup		Concurrency	Bulk Loading
		Insert Strategy	Structural Modification	Data Fitting Model	Position Search		
Updatable Learned Index	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
Traditional	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
	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

# 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

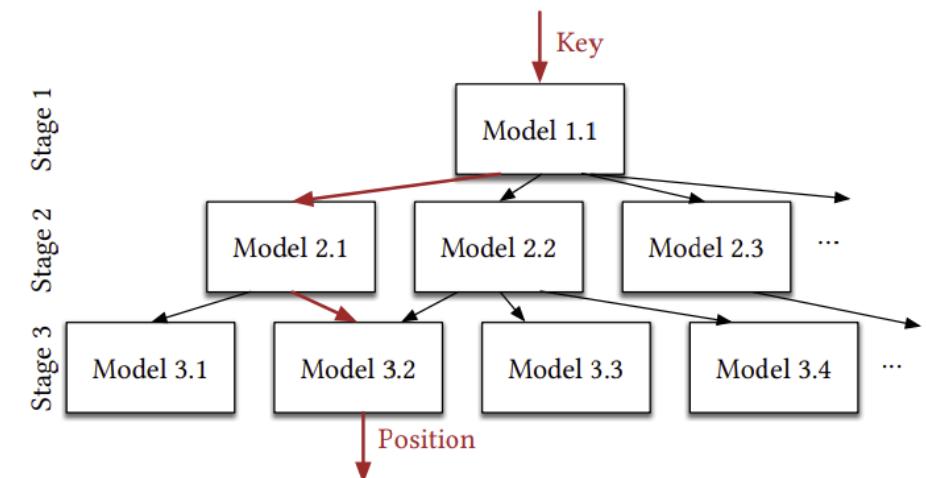
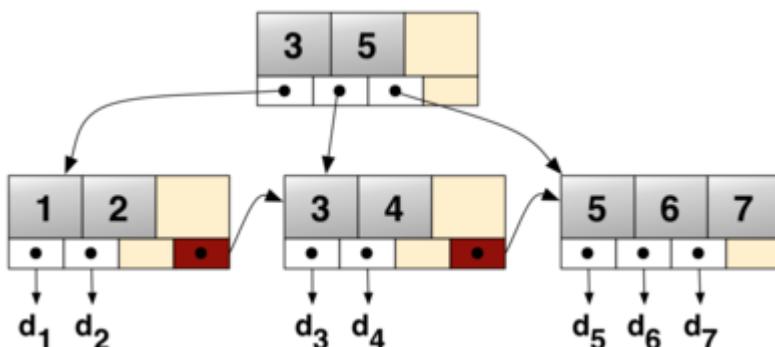


Figure 3: Staged models

# ALEX

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

Legend

Internal Node

Data Node

Key

Gap

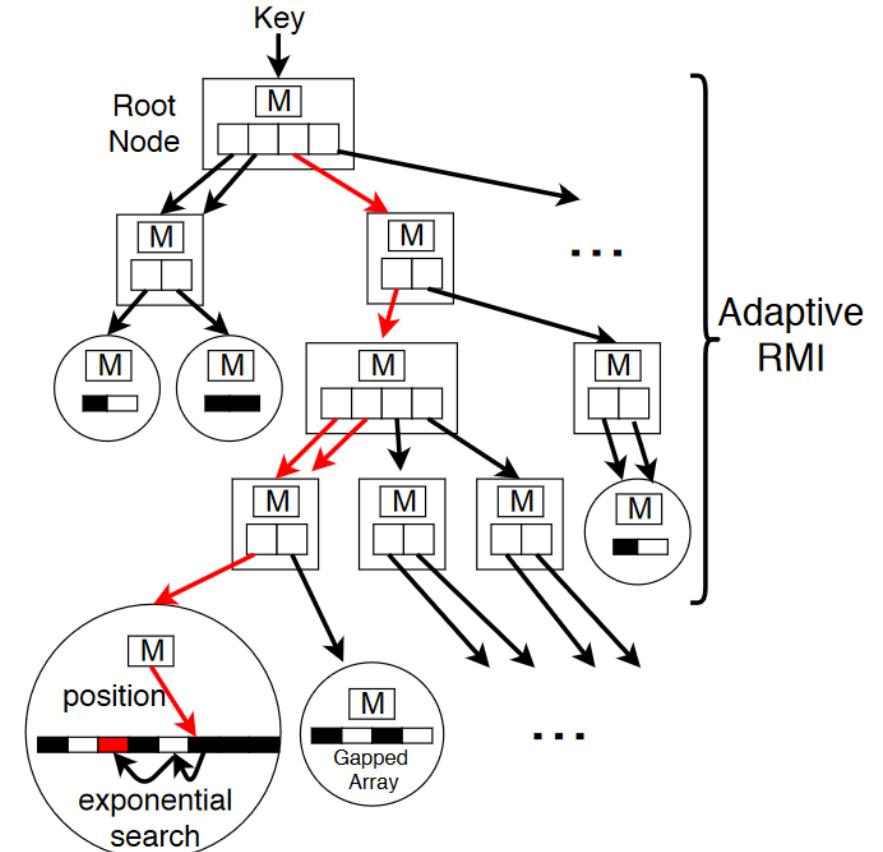


Figure 2: ALEX Design

# 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

1. Dense Array



Insertion Time

$O(n)$

2. B+ Tree Array



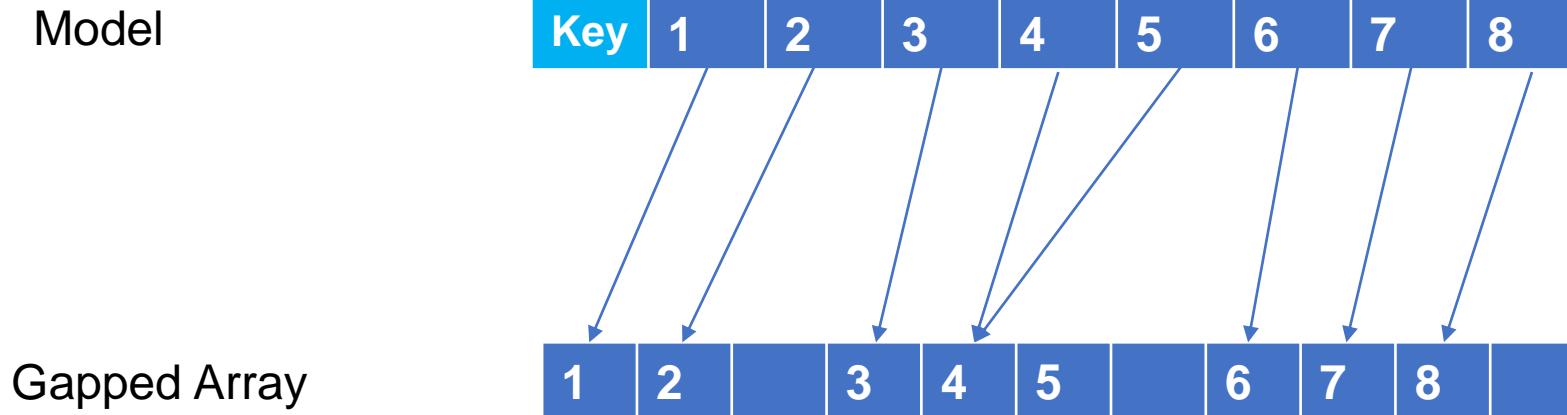
$O(n)$

3. Gapped Array

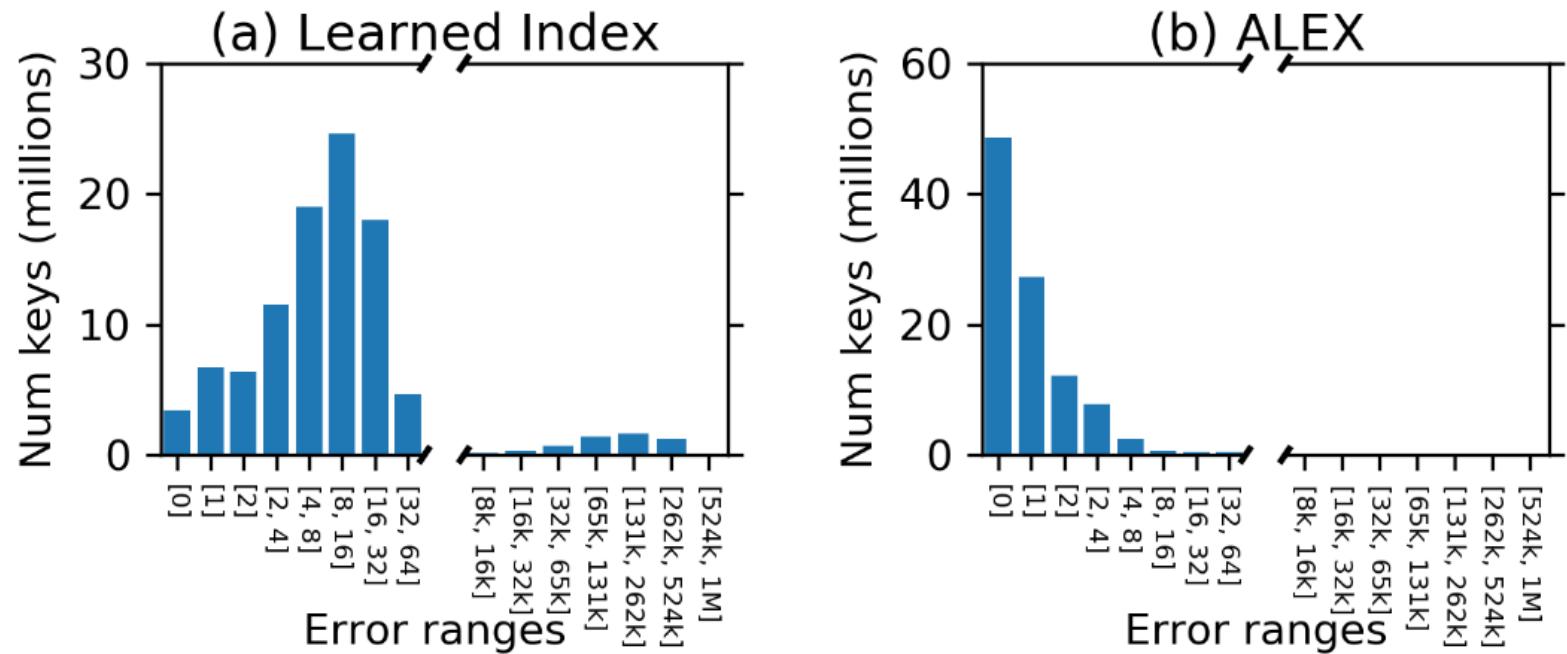


$O(\log n)$

# Idea 2. Model-based Inserts

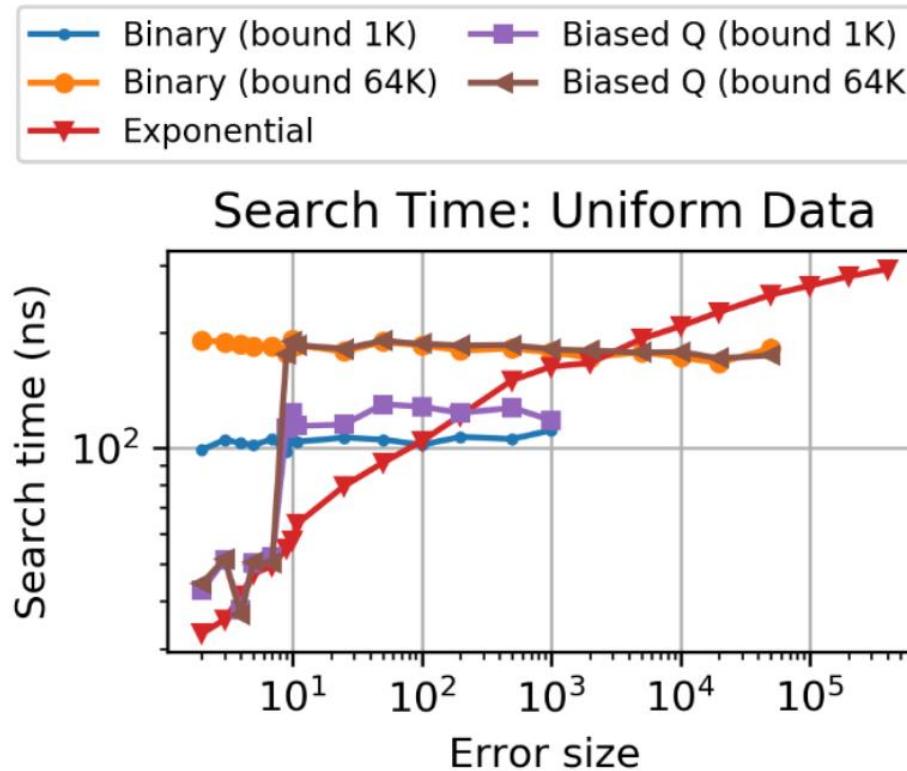


# Idea 2. Model-based Inserts



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

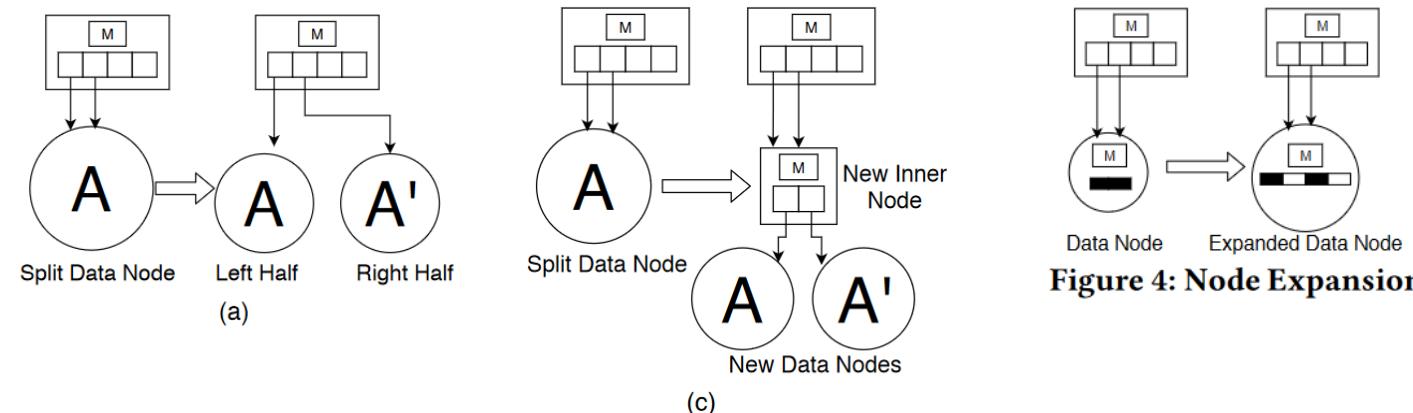


Figure 4: Node Expansion

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

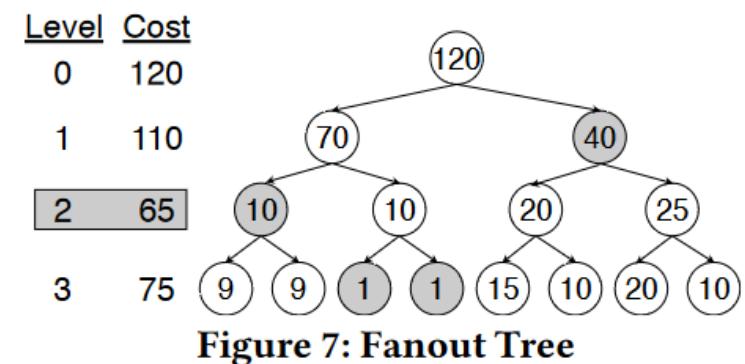


Figure 7: Fanout Tree

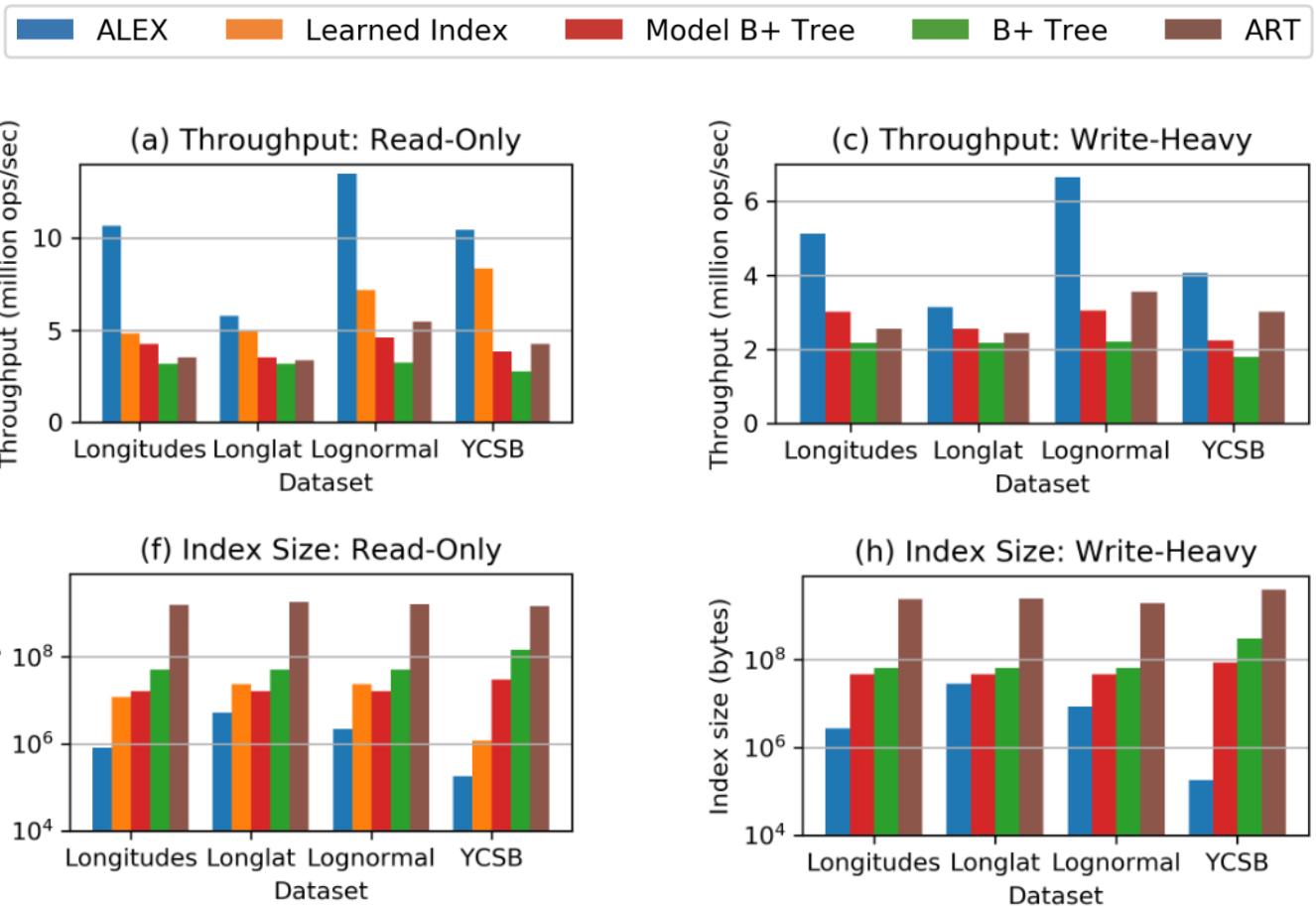
# Evaluation

## ■ High-level results

- Fast reads
- Fast writes
- Smaller index size

## ■ Other results

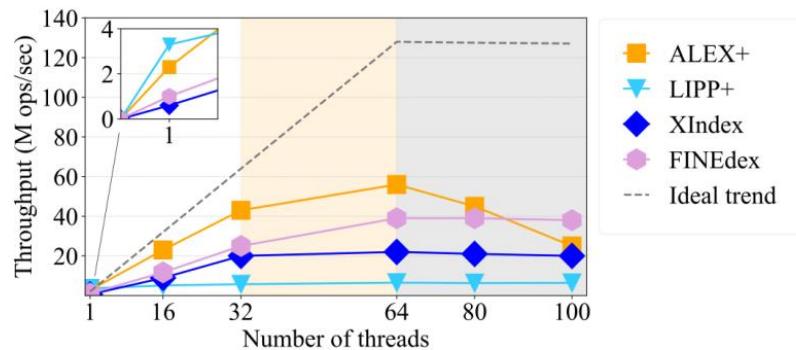
- Efficient bulk loading
- Scales
- Robust to data and workload shift



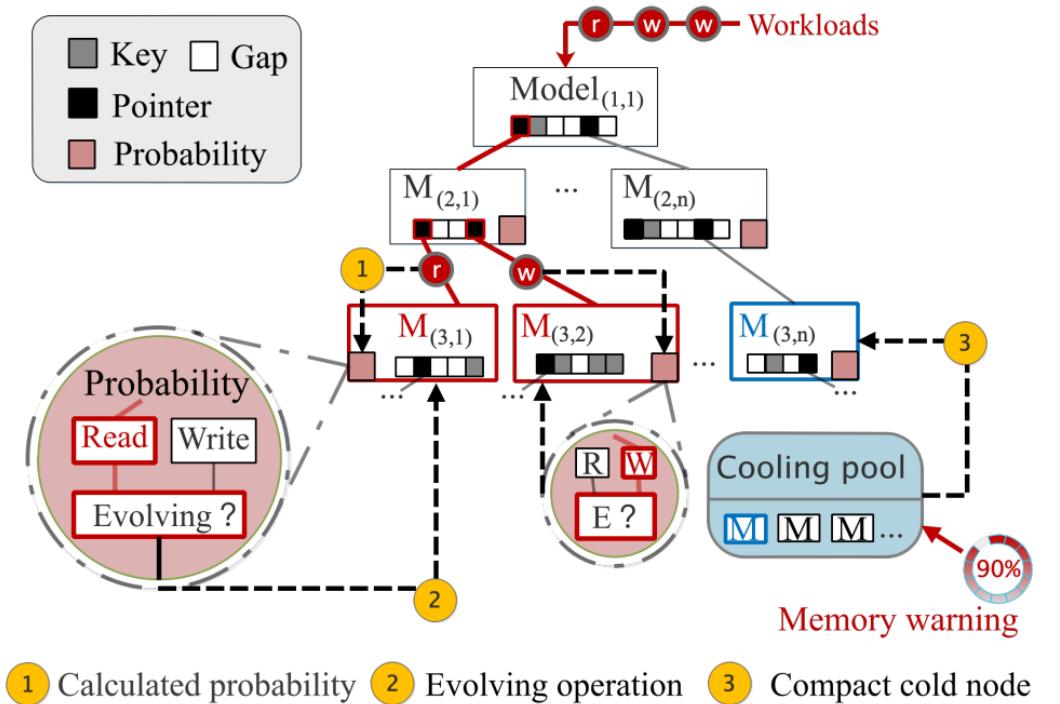
# Current Focus & Trends: Scalability

**Table 1: The limited scalability of existing schemes.**

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



**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.**

1. What is learned index?
2. Read-only learned indexes
3. Applying sampling for learned indexes
4. Updatable learned indexes
5. Research Topics
6. Tips & Tools

## 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.

## 🔗 Benchmarks

- Traditional Index : [Index-Microbench](#)
- Read-Only Learned Index : [LIST](#)
- Updatable Learned Index : [GRE \(with SALI\)](#)

## 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](#)"

# Lectures

CMU Intro to Database Systems (15-445/645 - Fall 2023)  
CMU Database Group  
https://15445.courses.cs.cmu.edu/fall2023/intro-to-database-systems-fall-2023/carnegie-mellon-university

F2023 #06 - Database Memory & Disk I/O Management (CMU Intro to Database Systems)  
CMU Database Group • 조회수 4.9천회 • 3개월 전

F2023 #07 - Hash Tables (CMU Intro to Database Systems)  
CMU Database Group • 조회수 4.5천회 • 3개월 전

F2023 #08 - B+Tree Indexes (CMU Intro to Database Systems)  
CMU Database Group • 조회수 4.5천회 • 3개월 전

F2023 #09 - Index Concurrency Control (CMU Intro to Database Systems)  
CMU Database Group • 조회수 3.4천회 • 3개월 전

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



비정형 빅데이터를 위한 키-밸류 DB  
최종무 | 상시모집 | 학습 기간 2021.12.01 ~ 2030.12.31

강좌 언어  
한국어

강의 길이  
7 주

권장 학습 시간  
4 시간

이수증 제공 여부  
미제공

신청현황 46 명 / 무제한

# 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 struggling 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 !

1. What is learned index?
2. Read-only learned indexes
3. Applying sampling for learned indexes
4. Updatable learned indexes
5. Research Topics
6. Tips & Tools

## 6. Tips & Tools

# 6.1. For Beginners

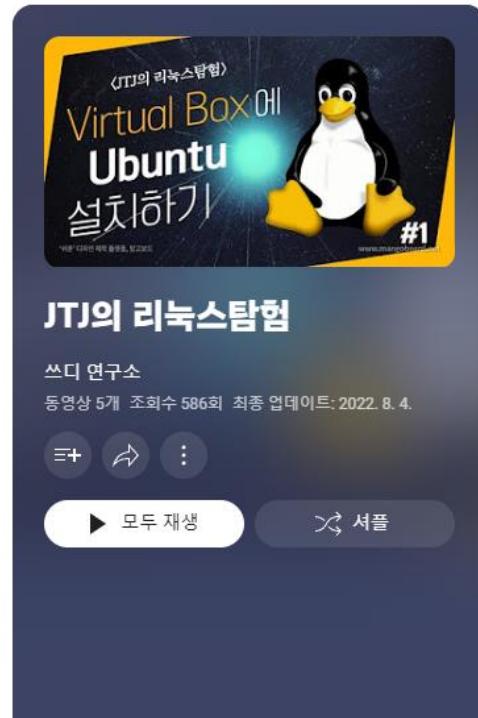
- As a computer system researcher, basic things you should know:
  - Environment Setup & Experimentation: *linux, makefile, vim, vscode*
  - Code Analysis: *gdb, utrace, understand*
  - Experimentation and Graphing: *shell programming, matplotlib*
  - How to Read Papers: S. Keshav, "How to Read a Paper"
  - 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*

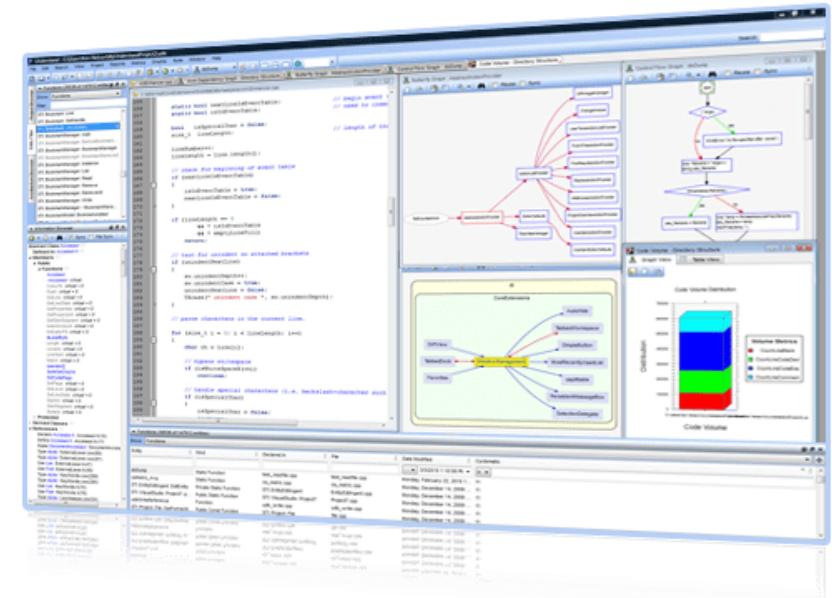
### ☞ For Beginners

- 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, "개발연구/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*

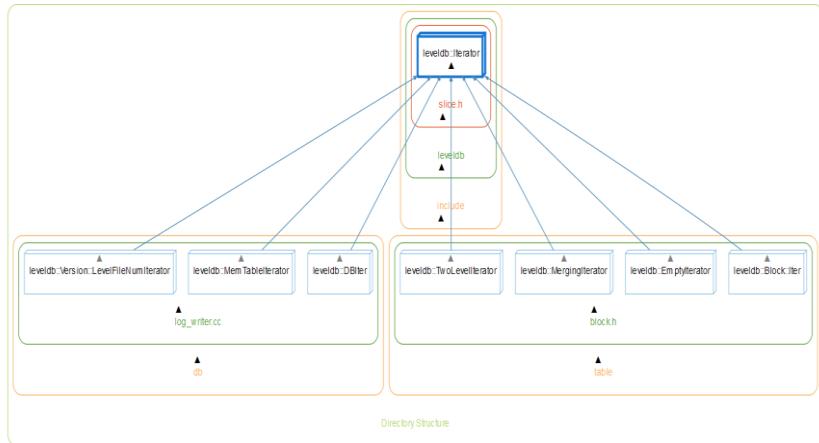
- Understand by SciTools
  - <https://www.scitools.com/> (ENG.)
  - <https://www.slexn.com/understand/> (KOR.)
- Static analysis tool
  - understanding of complex open-source code
  - Graphing
    - code dependencies, code flows, function calls, and more
- How to install
  - Register with university email
    - <https://licensing.scitools.com/register>
  - Free trial link
    - <https://www.scitools.com/pricing>
  - Download link
    - <https://licensing.scitools.com/download>



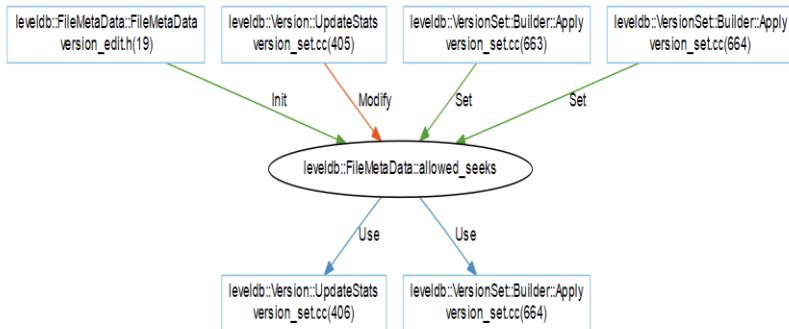
## Free for Educational Use

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](#)

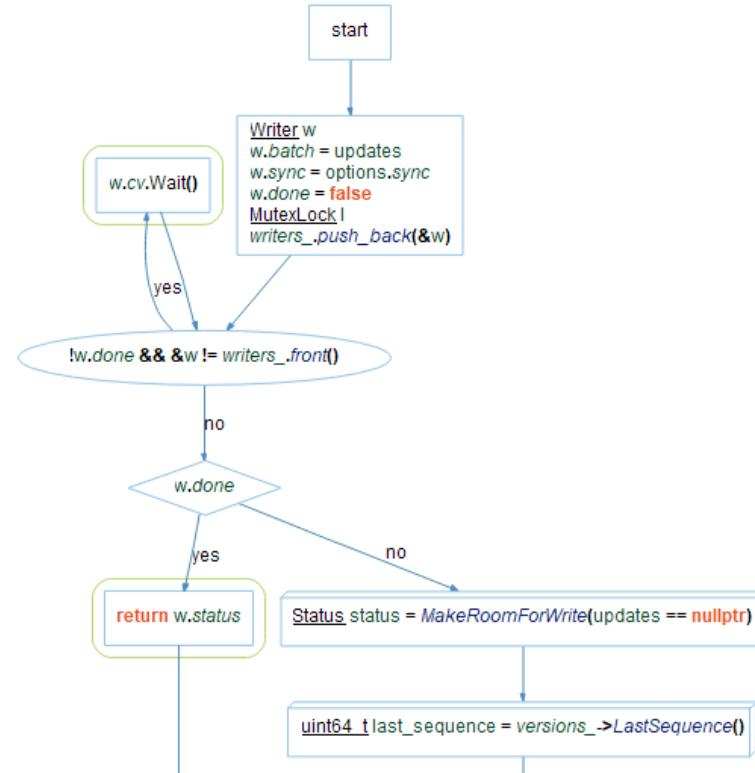
## 6.3.1. Code Analysis: *understand*



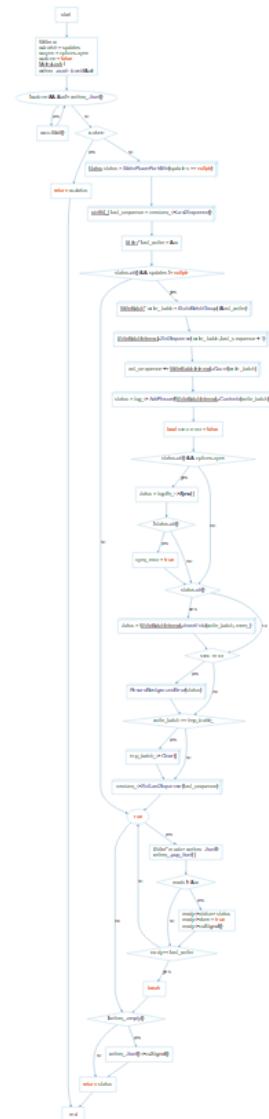
# UML Class Diagram: Iterator



## Object References

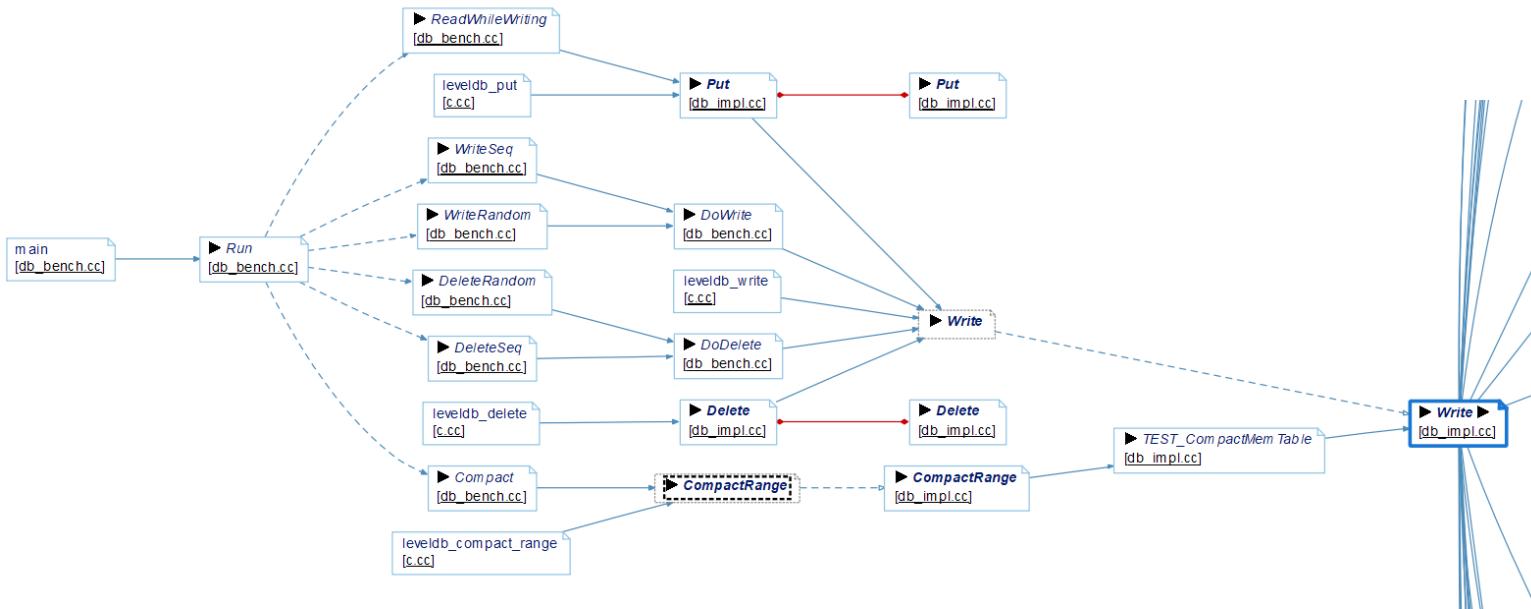


## Function Flow

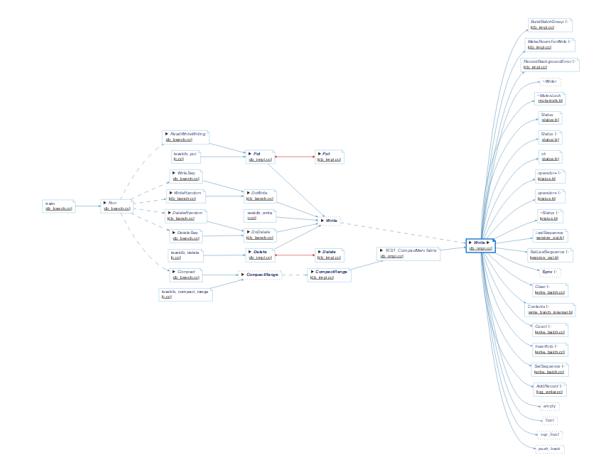


# 6.3.1. Code Analysis: understand

- Annotations ►
- Called By Depth ►
- Calls Depth ►
- Comments ►
- Edge Labels ►
- Filename ►
- Function Pointer ►
- Global Objects ►
- Layout ►
- Name ►
- Overrides ►
- Parameters ►
- Styled Labels ►
- Unresolved ►

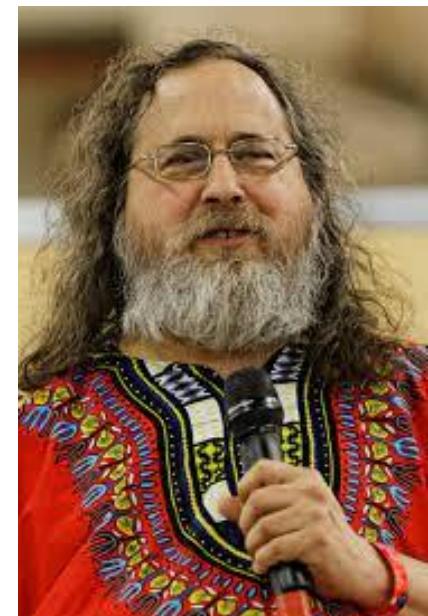


Butterfly: Function Calls & Called by



## 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.



## 6.3.2. Code Analysis: GDB

- LevelDB Example: Merge Iterator

```
1013      input->Next();
(gdb) s
leveldb::(anonymous namespace)::MergingIterator::Next(this=
 0x5555555590fee <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=0x5555555590f8c <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

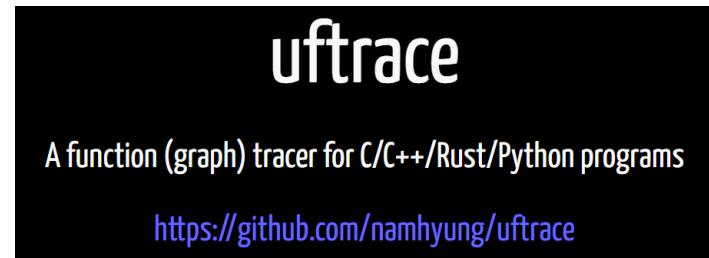
### ■ LevelDB Example: Merge Iterator

```
(gdb) display smallest
1: smallest = (leveldb::IteratorWrapper *) 0x0
(gdb) display child
2: child = (leveldb::IteratorWrapper *) 0x7ffffe8010438
(gdb) n
150      for (int i = 0; i < n_; i++) {
1: smallest = (leveldb::IteratorWrapper *) 0x7ffffe8010438
(gdb)
151      IteratorWrapper* child = &children_[i];
1: smallest = (leveldb::IteratorWrapper *) 0x7ffffe8010438
2: child = (leveldb::IteratorWrapper *) 0x7ffffe8010438
(gdb) display *smallest
3: *smallest = {iter_ = 0x7ffffe8012110, valid_ = true, key_ = {
   data_ = 0x7ffffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
(gdb) display *child
4: *child = {iter_ = 0x7ffffe8012110, valid_ = true, key_ = {
   data_ = 0x7ffffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
(gdb) n
152      if (child->Valid()) {
1: smallest = (leveldb::IteratorWrapper *) 0x7ffffe8010438
2: child = (leveldb::IteratorWrapper *) 0x7ffffe8010458
3: *smallest = {iter_ = 0x7ffffe8012110, valid_ = true, key_ = {
   data_ = 0x7ffffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
4: *child = {iter_ = 0x7ffffe800ff50, valid_ = true, key_ = {
   data_ = 0x7ffffe8012e30 '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->key()) < 0) {
                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

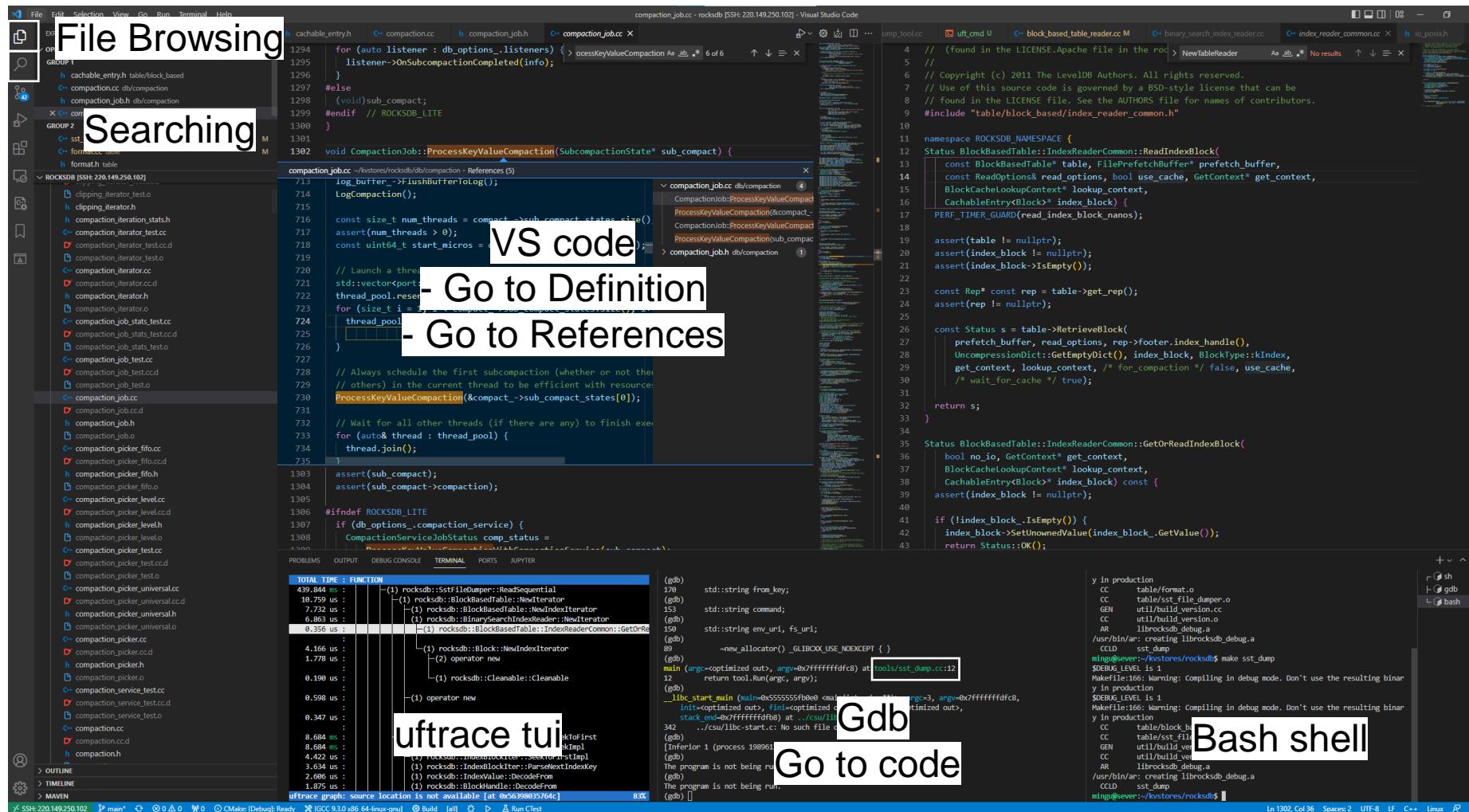
```
===== Back-trace ======
```

```
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 :             (48808728) 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 & utrace



# 6.4.1. Experimentation and Graphing : shell script

## Shell script

文 A 19 languages ▾

Article Talk

A **shell script** is a [computer program](#) designed to be run by a [Unix shell](#), a command-line interpreter.<sup>[1]</sup> 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](#).

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.

Read Edit View history Tools ▾



The screenshot shows a terminal window with a black background and white text. It displays a shell script named 'ipfirewall\_simple.sh'. The script contains several lines of code related to IP firewall configuration, including rules for 'simple\_inet' and 'simple\_out' interfaces. It includes comments like '# set these to your interface names' and '# Stop spending'. The script ends with a note '# Stop ipfirewall sets on the outside interface' and '#!/bin/sh'.

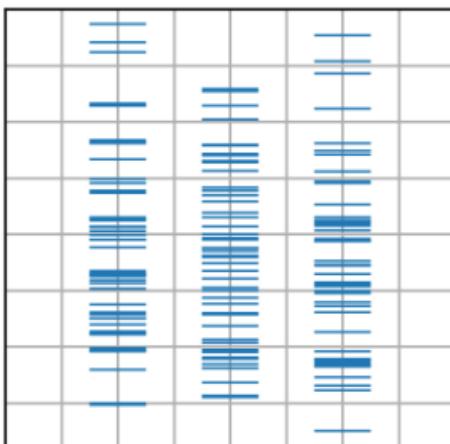
Editing a FreeBSD shell script for configuring ipfirewall

# 6.4.1. Experimentation and Graphing : shell script



- 1 [따배셀] 0. 따라하면서 배우는 Shell Programming 소개 영상!  
TTABAE-LEARN · 조회수 1.1만회 · 2년 전
- 2 [따배셀] 1. Linux Shell이란? (+ Shell 구성 실습)  
TTABAE-LEARN · 조회수 1.1만회 · 2년 전
- 3 [따배셀] 2. Bash shell과 변수  
TTABAE-LEARN · 조회수 7.1천회 · 2년 전
- 4 [따배셀] 3. Bash shell과 Rules  
TTABAE-LEARN · 조회수 6.6천회 · 2년 전
- 5 [따배셀] 4. Bash shell과 Rules 2  
TTABAE-LEARN · 조회수 4.3천회 · 2년 전

## 6.4.2. Experimentation and Graphing : *matplotlib*



### Matplotlib: Visualization with Python

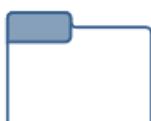
Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- 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

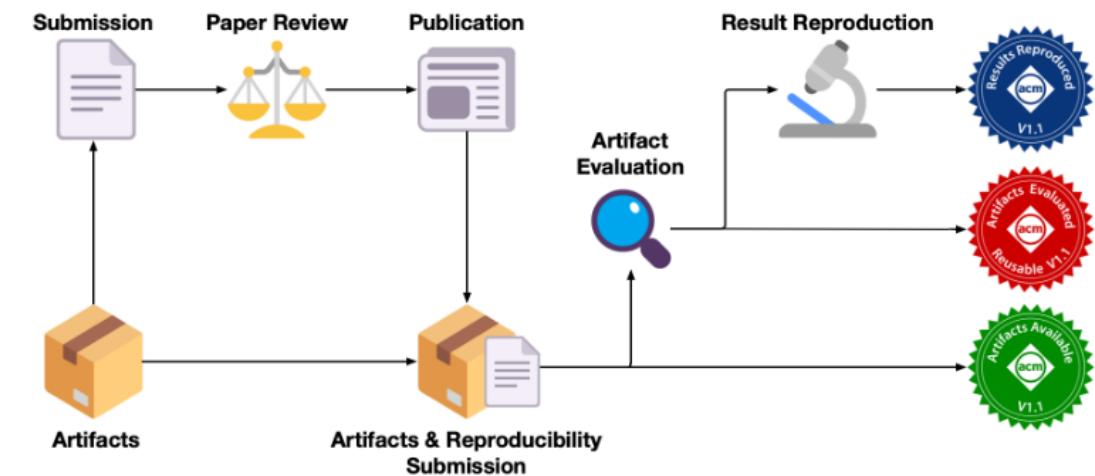
Why should I use matplotlib and not excel?



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.

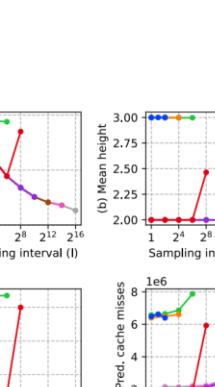
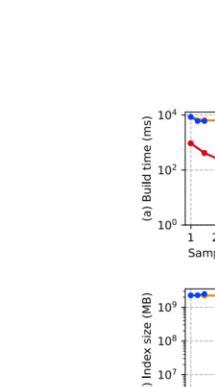
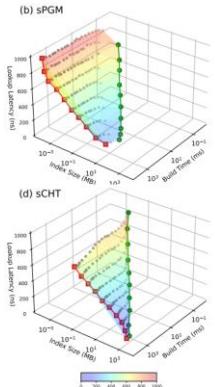
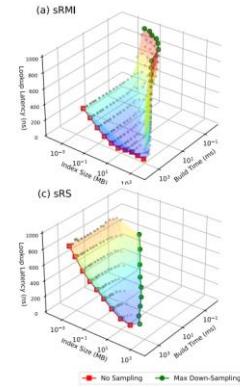


Figure 11: Design Space of indexing schemes on history dataset.

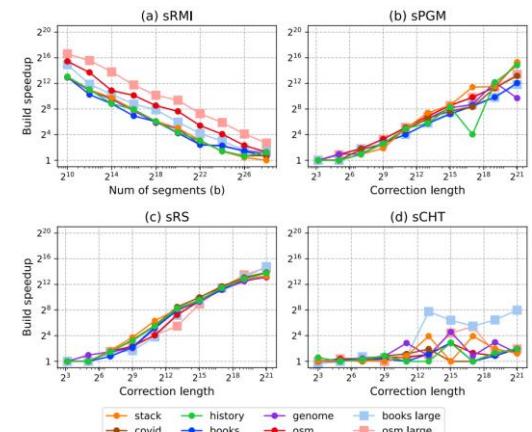


Figure 10: Build speedup with the safe interval.

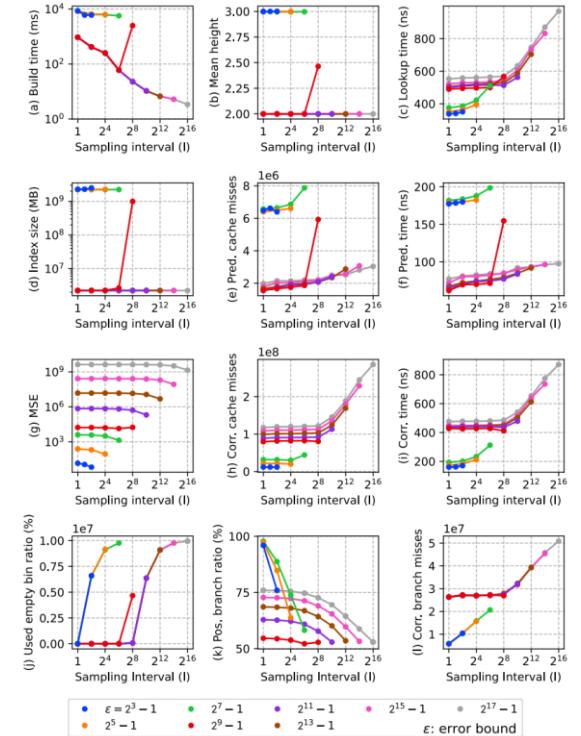


Figure 9: Performance impact of sampling on sCHT on history dataset.

# 6.4.2. VS Code with matplotlib & csv

Jupyter Notebook

```

if i != 0:
    axes[j,i].set_yticklabels([]) "yticklabels": Unknown word.
else:
    axes[j,i].set_yticklabels(tail_yvalues) "yticklabels": Unknown word.

lines, labels = fig.axes[-1].get_legend_handles_labels()
lines = lines[:1]
labels = labels[:1]

# new_order = [2, 1, 0, 3, 4, 5, 6, 7, 8]
# lines = [lines[i] for i in new_order]
# labels = [labels[i] for i in new_order]

plt.subplots_adjust(left=0.1, right=0.9, bottom=0.1, top=0.9, wspace=0.15, hspace=0.475) "wspace": Unknown
plt.savefig("avg_pareto.svg", bbox_inches="tight")

```

Matplotlib

```

col_dic = {"sRS":4, "sCHT":6, "sPGM":2, "sRMI(NB)":0,
           "ART": 12, "BTree": 8, "IBTree": 10, "sRT":16}

tab20 = plt.get_cmap('tab20') "cmap"
tab20c = plt.get_cmap('tab20c') "cmap"

traditional = ["BinarySearch", "BTree", "IBTree", "ART"]
rt = ["sRT", "sCHT"]
learned = ["sRMI(NB)", "sPGM", "sRS"]

groups = traditional + rt + learned

Plot2DPareto('search', 6.25, groups)

```

CSV File Rainbow

index,options,space,build,search type,dataset,search_0,search_50,search_99,search_99.9,s
1 ART,1048576,4952,24479,BinarySearch,books_200M_uint64,0.0,267.0,1351.0,1789.0,2162.0,5458.0,9656.0
2 ART,262144,13920,94201,BinarySearch,books_200M_uint64,0.0,264.0,1237.0,1668.0,1992.0,5090.0,25928.0
3 ART,65536,88704,315429,BinarySearch,books_200M_uint64,0.0,126.0,1096.0,1412.0,1772.0,4781.0,33264.0
4 ART,16384,188568,1065500,BinarySearch,books_200M_uint64,0.0,146.0,320.0,1308.0,1647.0,2039.0,11263
5 ART,4096,956496,5039929,BinarySearch,books_200M_uint64,0.0,140.0,833.0,1217.0,1543.0,1903.0,8216.0
6 ART,1024,3272616,19778021,BinarySearch,books_200M_uint64,0.0,115.0,691.0,1037.0,1332.0,1771.0,33221
7 ART,256,22360376,53829987,BinarySearch,books_200M_uint64,0.0,119.0,591.0,898.0,1147.0,1515.0,51024
8 ART,64,49097792,126209411,BinarySearch,books_200M_uint64,0.0,62.0,842.0,848.0,1088.0,1330.0,44455.0
9 ART,16,261696176,719184733,BinarySearch,books_200M_uint64,0.0,99.0,518.0,828.0,1060.0,1422.0,10261
10 BTree,1048576,3712,14010,BinarySearch,books_200M_uint64,0.0,277.0,1358.0,1798.0,2185.0,5616.0,1091
11 BTree,262144,14608,46843,B1
12 BTree,65536,57096,15378,B1
13 BTree,16384,227048,490561,B
14 BTree,4096,906864,1363692,BinarySearch,books_200M_uint64,0.0,143.0,758.0,1190.0,1495.0,4662.0,206
15 BTree,1024,3626672,3754759,BinarySearch,books_200M_uint64,0.0,177.0,656.0,1025.0,1309.0,1787.0,84
16 BTree,256,14502616,14923662,BinarySearch,books_200M_uint64,0.0,177.0,656.0,1025.0,1309.0,1787.0,84
17 BTree,64,58008568,49839674,BinarySearch,books_200M_uint64,0.0,144.0,676.0,985.0,1220.0,1621.0,8647
18 BTree,16,232032392,210538207,BinarySearch,books_200M_uint64,0.0,178.0,616.0,920.0,1150.0,1413.0,85
19 BinarySearch,0,0,BinarySearch,books_200M_uint64,0.0,161.0,957.0,1324.0,1644.0,2878.0,27381.0
20 IBTree,1048576,4128,10930,BinarySearch,books_200M_uint64,0.0,320.0,1374.0,1813.0,2172.0,5338.0,106
21 IBTree,262144,16472,45360,BinarySearch,books_200M_uint64,0.0,266.0,1242.0,1672.0,2028.0,4836.0,607
22 IBTree,65536,53480,145873,binarySearch,books_200M_uint64,0.0,161.0,1866.0,1468.0,1819.0,5163.0,502
23 IBTree,16384,201512,528372,BinarySearch,books_200M_uint64,0.0,159.0,392.0,1334.0,1688.0,2037.0,469
24 IBTree,4096,789528,1597192,BinarySearch,books_200M_uint64,0.0,166.0,817.0,1194.0,1530.0,2066.0,381

CSV File Editor

column 1	column 2	column 3	column 4	column 5	column 6	column 7
110 sRMI(NB)	256.0/000000/65536	4128	178212	BinarySearch	books_200M_uint64	0.0
111 sRMI(NB)	256.0/000000/16384	4128	675179	BinarySearch	books_200M_uint64	0.0
112 sRMI(NB)	1024.0/000000/16384	16416	676719	BinarySearch	books_200M_uint64	0.0
113 sRMI(NB)	4096.0/000000/4096	65568	2180415	BinarySearch	books_200M_uint64	0.0
114 sRMI(NB)	16384.0/000000/1024	262176	6048374	BinarySearch	books_200M_uint64	0.0
115 sRMI(NB)	16384.0/000000/256	262176	21412351	BinarySearch	books_200M_uint64	0.0
116 sRMI(NB)	65536.0/000000/256	1048608	22487105	BinarySearch	books_200M_uint64	0.0
117 sRMI(NB)	65536.0/000000/64	1048608	70755295	BinarySearch	books_200M_uint64	0.0
118 sRMI(NB)	262144.0/000000/64	4194326	72969247	BinarySearch	books_200M_uint64	0.0
119 sRMI(NB)	1048576.0/000000/1	1943304	0.0/000000/1	BinarySearch	books_200M_uint64	0.0
120 sRMI(NB)	1048576.0/000000/1	1943304	1943304	BinarySearch	books_200M_uint64	0.0
121 sRMI(NB)	4194304.0/000000/1	1943304	1943304	BinarySearch	books_200M_uint64	0.0
122 sRMI(NB)	4194304.0/000000/4	67108896	822089465	BinarySearch	books_200M_uint64	0.0
123 sRMI(NB)	16777216.0/000000/4	268435488	974613327	BinarySearch	books_200M_uint64	0.0
124 sRMI(NB)	67108864.0/000000/4	1073741856	1130750542	BinarySearch	books_200M_uint64	0.0
125 sRMI(NB)	4194304.0/000000/1	67108896	2874910153	BinarySearch	books_200M_uint64	0.0
126 sRMI(NB)	67108864.0/000000/1	1073741856	3490739585	BinarySearch	books_200M_uint64	0.0
127 sRMI(NB)	67108864.0/000000/1	1073741856	4010482648	BinarySearch	books_200M_uint64	0.0
128 sRS	6/1/262144/524290/761	12416	59445	BinarySearch	books_200M_uint64	0.0
129 sRS	6/1/65536/131074/3036	48816	180041	BinarySearch	books_200M_uint64	0.0
130 sRS	10/1/65536/131074/3036	50736	184433	BinarySearch	books_200M_uint64	0.0
131 sRS	8/1/16384/32770/12107	194336	704945	BinarySearch	books_200M_uint64	0.0
132 sRS	10/1/16384/32770/12107	195872	708763	BinarySearch	books_200M_uint64	0.0
133 sRS	12/1/16384/32770/12107	202016	721342	BinarySearch	books_200M_uint64	0.0
134 sRS	14/1/4096/8194/48055	801760	2383692	BinarySearch	books_200M_uint64	0.0

# 6.5. How to Read Papers

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.

Cited by	VIEW ALL	
	All	Since 2019
Citations	20517	3357
h-index	63	27
i10-index	169	72



**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



젊은 과학자에게  
Peter Medawar | 조호근 옮김

The intensity of  
the CONVICTION  
that A HYPOTHESIS  
is TRUE  
has NO BEARING on  
whether it IS  
TRUE or NOT.



대학원이라는 미지의 영역에 대한 궁금의 안내서!

- 블로그 방문 170만 회!
- 페이스북 3만 명 팔로워 5만 3,000회 공유!
- 슬라이드 뷰어 60만 뷰!



대학원 선배들이 이끼는 후배에게 해주는 현실적인 조언!

- 대입종 최윤석 박사 추천
- 알쓸신잡 3 김성욱 교수 추천
- 대학원 진학 전 필독서!



Remzi Arpacı-Dusseau

University of Wisconsin-Madison  
Verified email at cs.wisc.edu - [Homepage](#)  
[Operating Systems](#)

Graduate School  
KEYS TO SUCCESS  
(in research)

Remzi H. Arpacı-Dusseau  
University of Wisconsin-Madison

## 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 있을지도 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](#)

## 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!