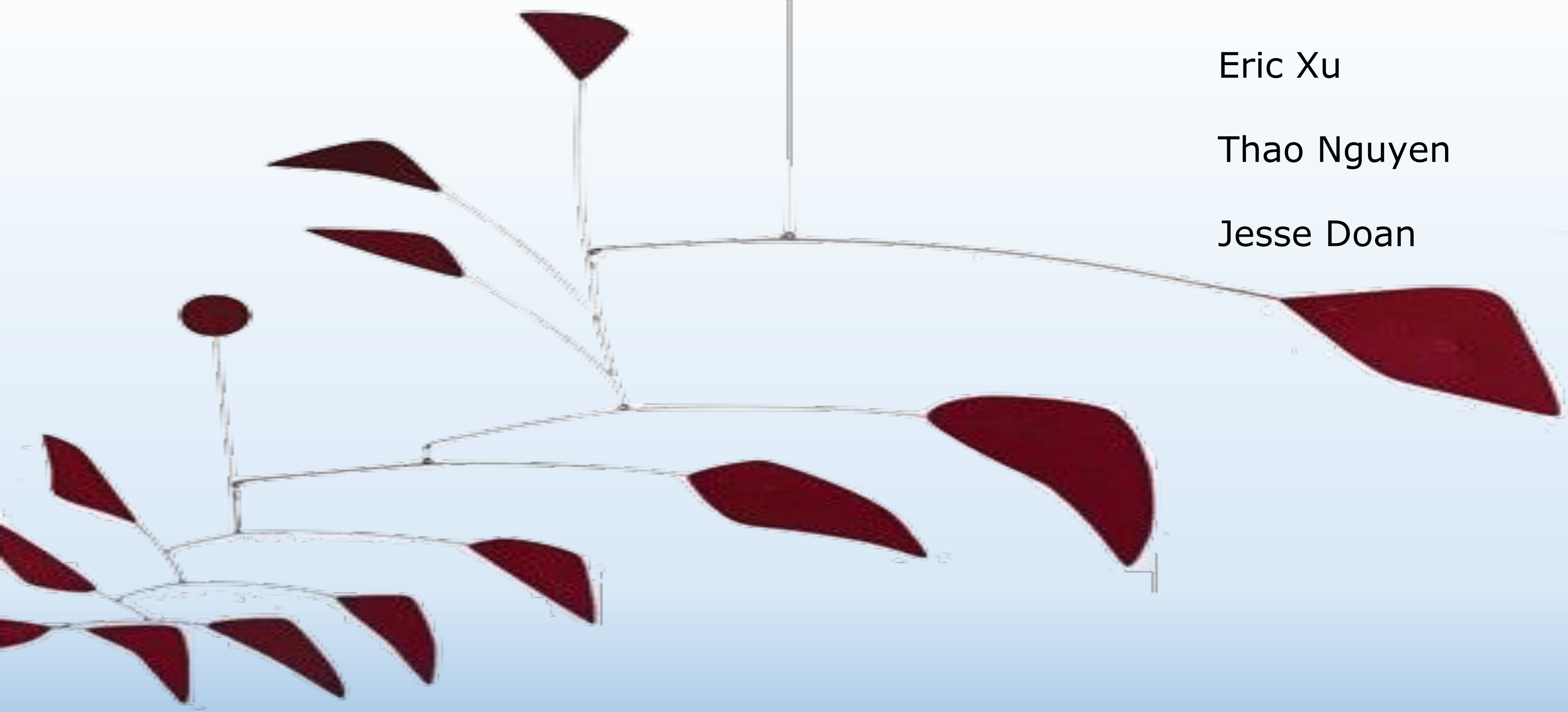
Eric Xu

Thao Nguyen

Jesse Doan



CLRS - An Introduction to Learned Hash Index

Bucket List

- ❑ Introduction
- ❑ Previous Work
- ❑ Motivation
- ❑ Calibrated Linear Recursive Structure (CLRS)
- ❑ Results
- ❑ Conclusion & Future Work

Bucket List

- ❏ **Introduction**
- ❏ Previous Work
- ❏ Motivation
- ❏ Calibrated Linear Recursive Structure (CLRS)
- ❏ Results
- ❏ Conclusion & Future Work

Learned Index...what's that?

- Problem: traditional data structures don't account for properties of data
- Index can be thought of as a mapping from key to value
- Use machine learning to overfit the underlying data distribution
- Structured data inputs: mapping from keys to values is easy to learn
- Benefits of customized indexes:
  - Scale with complexity, not with data
  - Leverage specialized hardware & parallelism for fast inference
  - Space saving: think data compression

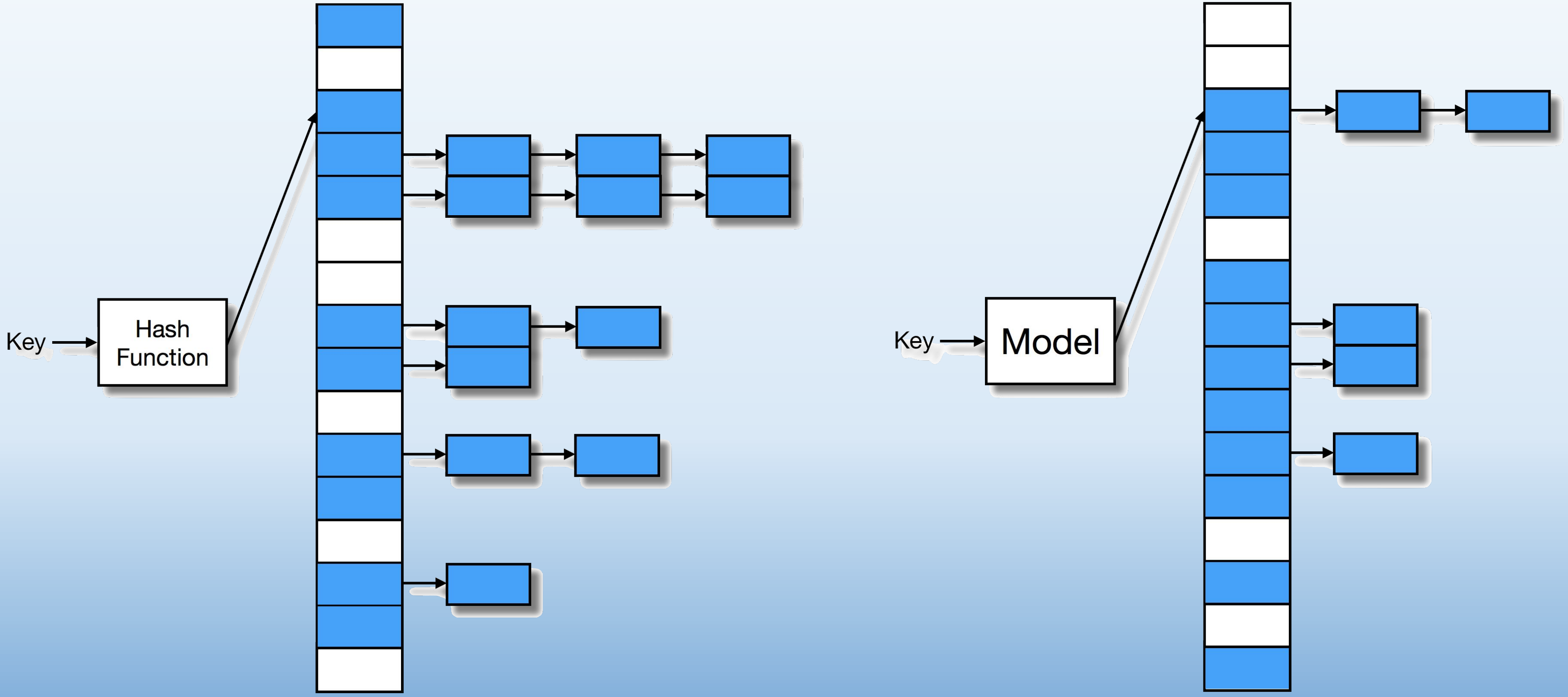
Bucket List

- ✓ Introduction
- ❑ **Previous Work**
- ❑ Motivation
- ❑ Calibrated Linear Recursive Structure (CLRS)
- ❑ Results
- ❑ Conclusion & Future Work

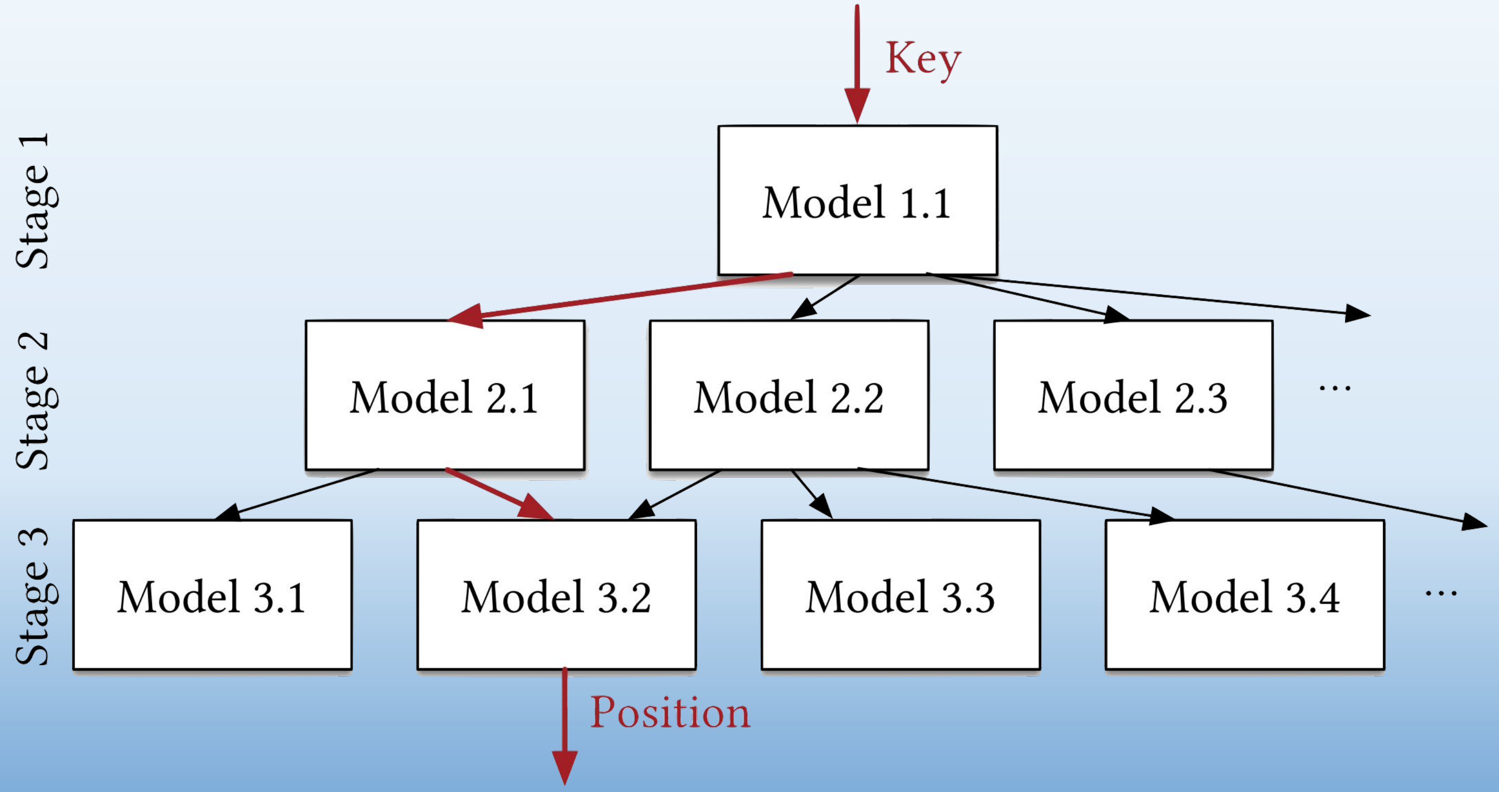
Previous Work

- The Case for Learned Index Structures, Kraska et al.
- Learned model to solve 3 kinds of indexes:
  - Existence Index - Bloom Filter
  - Range Index - B-Tree
  - Point Index - Hash Function
- Fall back on traditional data structures for performance guarantees

Learned HashMap



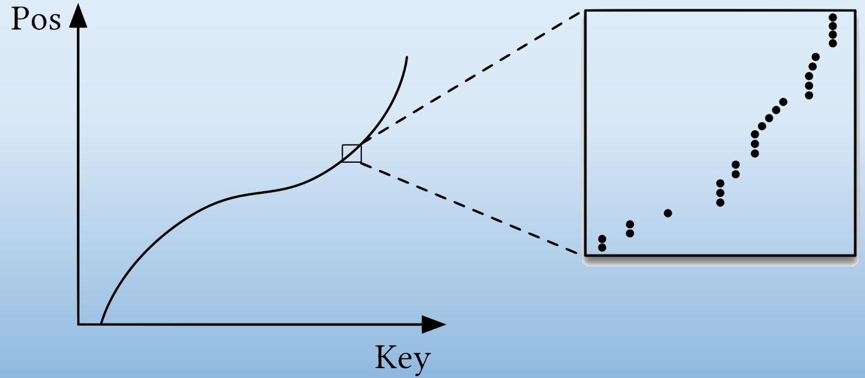
Last Mile Problem & Recursive Model Index (RMI)



Key Insight

Learned Index Structure...

- models the CDF of input data
- operates as a hash function $F: U \rightarrow [m]$
  - Element x hashes to bucket $F(x) \cdot m$
  - Potentially has no collisions



Bucket List

- ✓ Introduction
- ✓ Previous Work
- ❏ **Motivation**
- ❏ Calibrated Linear Recursive Structure (CLRS)
- ❏ Results
- ❏ Conclusion & Future Work

Motivating Questions

1. Can the learned hash function learn more complex CDF distributions?
   - Kraska et. al used on linear and lognormal datasets

Datasets

Synthetic datasets (size 100,000)

- Linear: uniform distribution in the interval (-5, 5)
  - CDF of a uniform distribution is linear
  - Should be easy for our model to learn
- Lognormal: log-normal distribution with $\mu = 0$ and $\sigma = 2$
  - Contains a heavy-tail, making the CDF very non-linear
  - Same distribution used by Kraska et. al
- Normal: normal distribution with $\mu = 0$ and $\sigma = 0.0001$
  - Extremely low variance causes the CDF to contain a large jump within a small range
  - Should be more difficult to learn

Constructing the Datasets

- Sort *N* sampled keys
- *i*th key maps to value *i*/*N*
  - CDF evaluated at the *i*th key is approximately *i*/*N*

Key	Value
-2.5	0.0
-1.2	0.2
0.3	0.4
1.4	0.6
3.6	0.8

Motivating Questions

1. Can the learned hash function learn more complex CDF distributions?
    - Kraska et. al used on linear and lognormal datasets

2. How does the performance of a learned hash function compare with traditional hash functions?
    - Kraska et. al primarily used collision rate as the comparison metric

Hash Performance Metrics

Comparison metrics across different hash functions:

- Collision Rate
  - Fraction of the utilized buckets with hash conflicts
- Bucket Utilization
  - Fraction of buckets containing at least 1 element
- Average Bucket Height
  - Equivalent to average runtime for accessing, inserting, or removing in hash-map

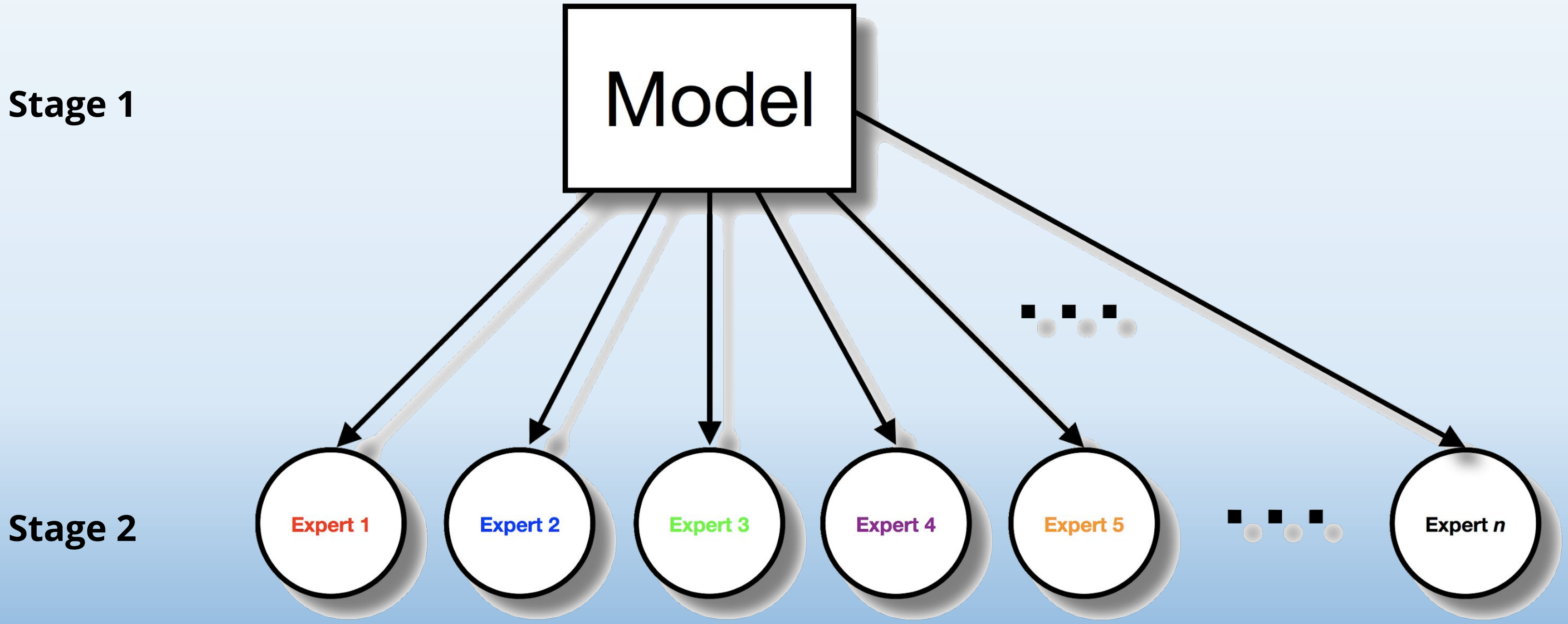
Other considerations for learned hash function:

- Memory/number of trained parameters
- Training time

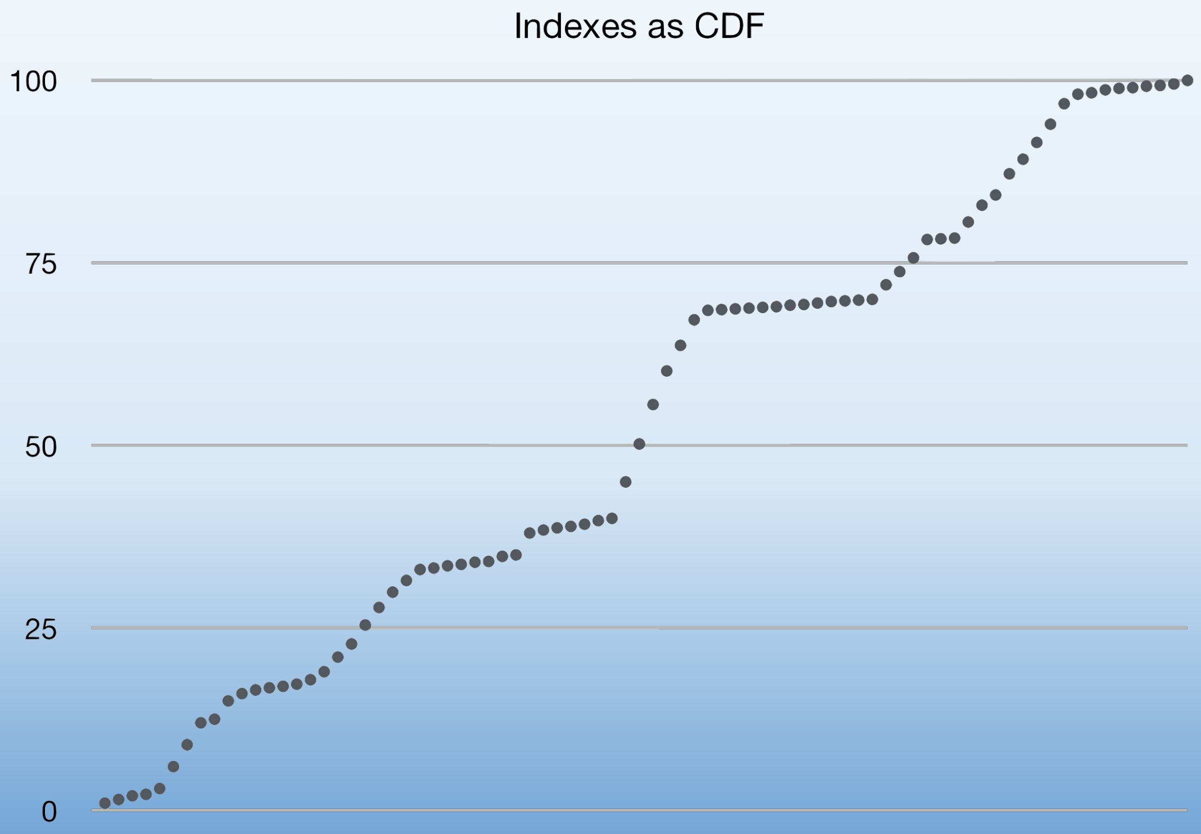
Bucket List

- ✓ Introduction
- ✓ Previous Work
- ✓ Motivation
- ❑ **Calibrated Linear Recursive Structure (CLRS)**
- ❑ Results
- ❑ Conclusion & Future Work

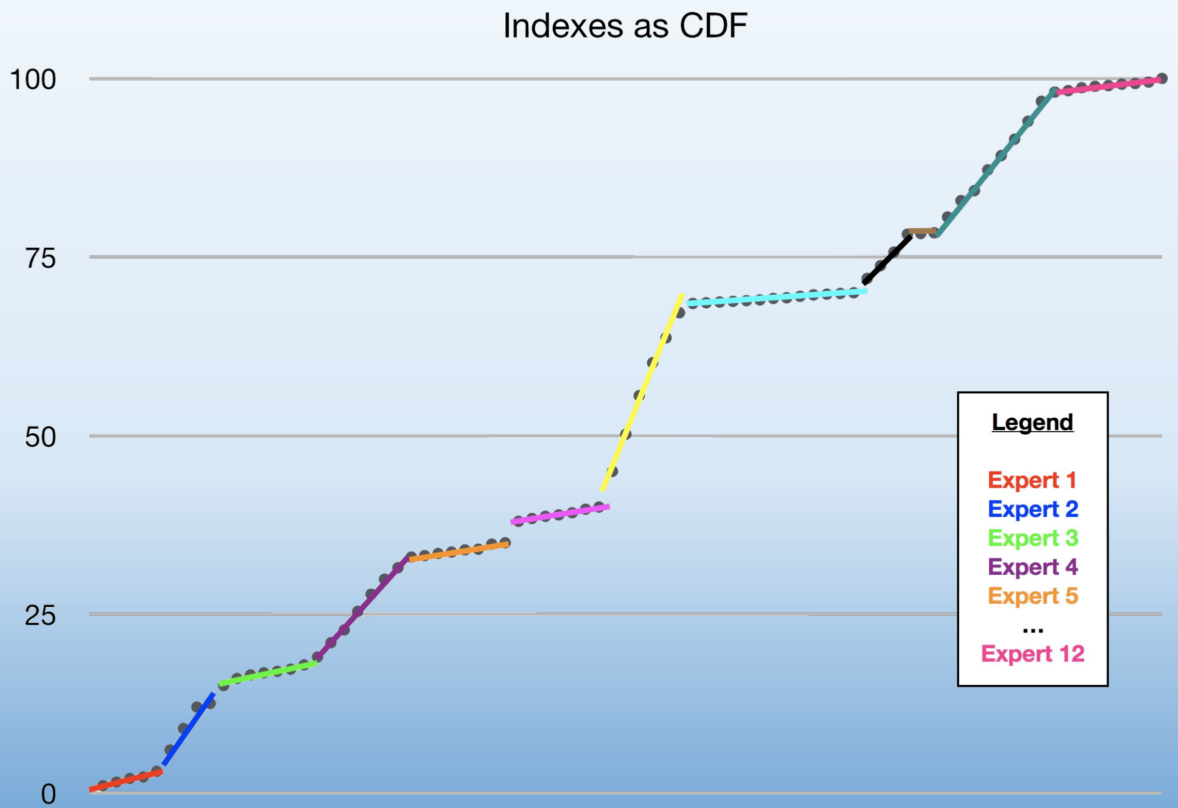
Calibrated Linear Recursive Structure (CLRS)



Calibrated Linear Recursive Structure (CLRS)



Calibrated Linear Recursive Structure (CLRS)



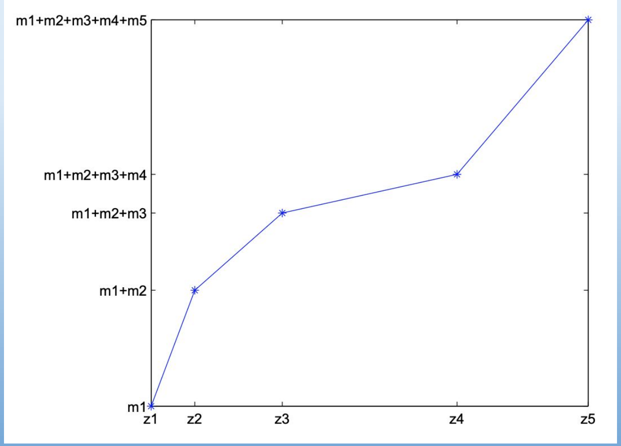
Calibrated Linear Recursive Structure (CLRS)

- Map inputs into some other space so that they are more spread out
- Take advantage of the monotonic relationship between keys and values
  - As keys increase, the values also increase

- Stage 1 Model and Stage 2 Experts are **Calibrated Linear Models**
- Enforces the monotonic constraint and outputs a prediction

Calibrated Linear Recursive Structure (CLRS)

- Model learns distribution by learning a piecewise linear function $\mathbf{\Phi}$
  - $\mathbf{\Phi}$ is determined by a set of knots
- Model calibrates input *x* by mapping it to a position $\mathbf{\Phi}(x)$
- $\mathbf{\Phi}$(x) is fed through a linear regression model to get prediction
- Total number of parameters is determined by number of knots used to parameterize $\mathbf{\Phi}$



Calibrated Linear Recursive Structure (CLRS)

Summary

- Calibrated Linear
  - Helps training by preprocessing inputs so that they are more spread out
  - Takes advantage of the monotonicity between keys and values
- Recursive Structure
  - Breaks prediction into 2 stages
  - Model in stage 1 sends off the input to an expert in stage 2 that makes a prediction

Bucket List

- ✓ Introduction
- ✓ Previous Work
- ✓ Motivation
- ✓ Calibrated Linear Recursive Structure (CLRS)
- ❑ **Results**
- ❑ Conclusion & Future Work

Baseline Results

load factor = 1

Data Set	Collision Rate	Bucket Util	Avg Bucket Height
linear.test	0.416	0.6341	1.577
lognormal.test	0.414	0.6350	1.575
normal.test	0.416	0.6312	1.584

Table 1: MD5 Performance

Data Set	Collision Rate	Bucket Util	Avg Bucket Height
linear.test	0.413	0.6328	1.580
lognormal.test	0.420	0.6310	1.585
normal.test	0.419	0.6307	1.586

Table 2: Murmur3 Performance

Theoretical Bounds

Let *h* be a uniform, *n*-independent hash function that hashes elements to *n* buckets. The expected fraction of non-empty buckets after hashing *n* elements is 1 - 1/e.

Proof Outline:

Focus on some bucket *i* and calculate the probability that at least one element hashes to bucket *i* : 1 - (1 - 1/n)^n

Use Linearity of Expectation and notice that as n increases, (1 - 1/n)^n approaches 1/e

This yields 1 - 1/e (or approximately 0.632), as desired

Baseline Results

Pretty close to 1 - 1/e (≈0.632)!

Data Set	Collision Rate	Bucket Util	Avg Bucket Height
linear.test	0.416	0.6341	1.577
lognormal.test	0.414	0.6350	1.575
normal.test	0.416	0.6312	1.584

Table 1: MD5 Performance

Data Set	Collision Rate	Bucket Util	Avg Bucket Height
linear.test	0.413	0.6328	1.580
lognormal.test	0.420	0.6310	1.585
normal.test	0.419	0.6307	1.586

Table 2: Murmur3 Performance

Theoretical Bounds

Let *h* be a uniform, *n*-independent hash function that hashes elements to *n* buckets. The expected fraction of buckets containing a collision after hashing *n* elements is at most 1 - 2/e.

Proof Outline:

Similar to earlier, focus on some bucket *i* and calculate the probability that at least two elements hash to bucket *i* : 1 - (1 - 1/n)^n - (1 - 1/n)^(n - 1)

Use Linearity of Expectation and notice that as n increases, (1 - 1/n)^n approaches 1/e

Upper bound (1 - 1/n)^(n - 1) to (1 - 1/n)^n, which yields 1 - 2/e (or approximately 0.264), as desired

Baseline Results

Collision Rate × Bucket Util = 0.41 × 0.63 = 0.2583, which is pretty close to 1 - 2/e (≈0.264)!

Data Set	Collision Rate	Bucket Util	Avg Bucket Height
linear.test	0.416	0.6341	1.577
lognormal.test	0.414	0.6350	1.575
normal.test	0.416	0.6312	1.584

Table 1: MD5 Performance

Data Set	Collision Rate	Bucket Util	Avg Bucket Height
linear.test	0.413	0.6328	1.580
lognormal.test	0.420	0.6310	1.585
normal.test	0.419	0.6307	1.586

Table 2: Murmur3 Performance

Learned Index Results

Data Set	Collision Rate	Bucket Util	Avg Bucket Height
linear.test	0.419	0.633	1.579
lognormal.test	0.419	0.631	1.584
normal.test	0.417	0.632	1.582

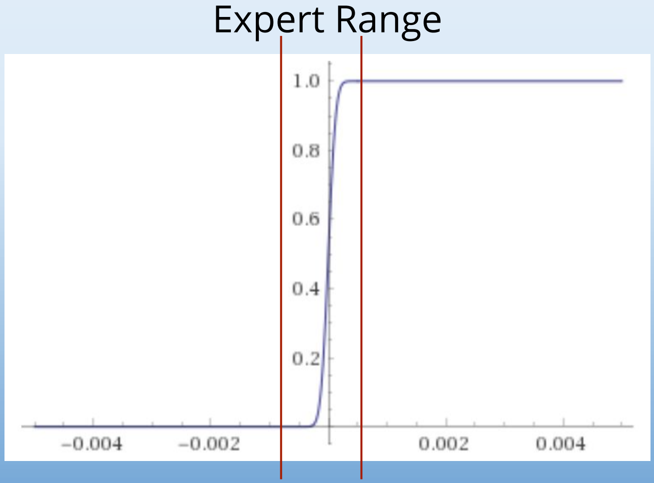
Table 3: Learned-Index Stage 1 Test Performance

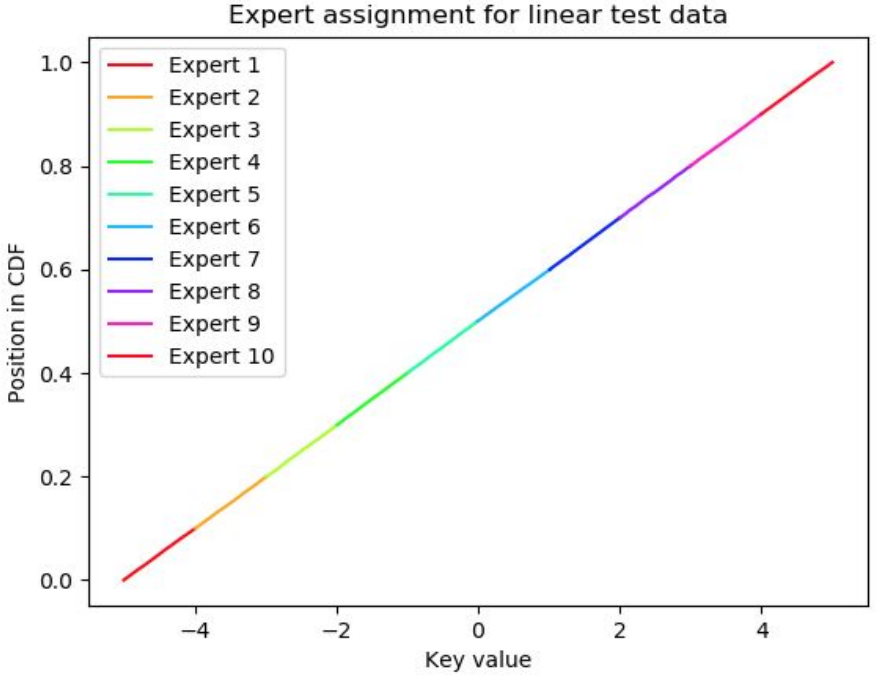
Data Set	Collision Rate	Bucket Util	Avg Bucket Height
linear.test	0.384	0.675	1.482
lognormal.test	0.394	0.666	1.503

Table 4: Learned-Index Stage 2 Test Performance

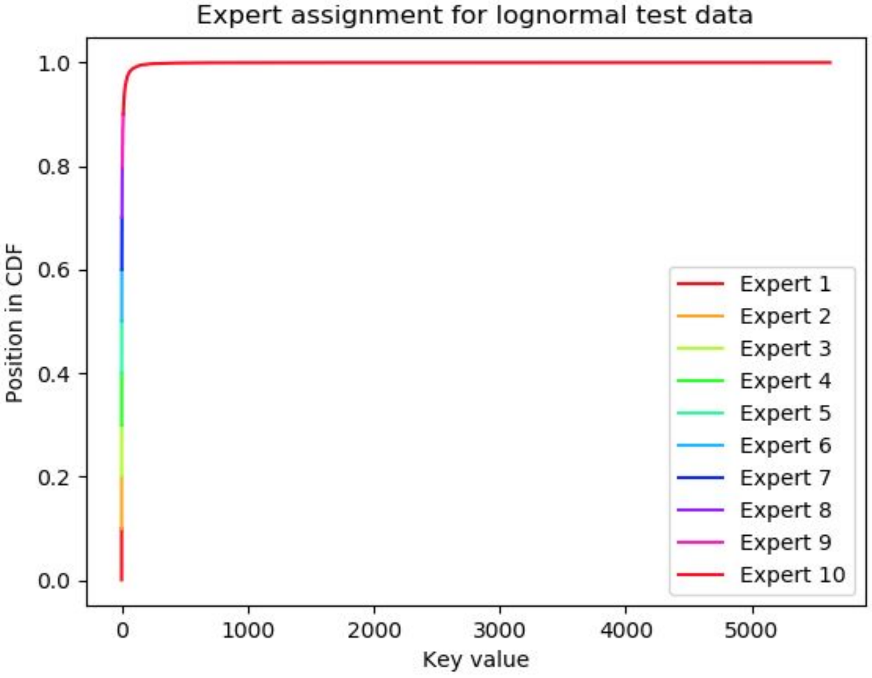
Analysis

- CLRS uses 10 experts and all Calibrated Linear Models use 1000 knots
  - Total number of trained parameters ≈ 11,000
- Models are trained for one epoch (2-3 min)
- Normal distribution performs poorly on stage 2





Expert assignment for linear test data
Position in CDF
Key value
Expert 1
Expert 2
Expert 3
Expert 4
Expert 5
Expert 6
Expert 7
Expert 8
Expert 9
Expert 10
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4



Expert assignment for lognormal test data
Position in CDF
Key value
Expert 1
Expert 2
Expert 3
Expert 4
Expert 5
Expert 6
Expert 7
Expert 8
Expert 9
Expert 10
1.0
0.8
0.6
0.4
0.2
0.0
0
1000
2000
3000
4000
5000

Bucket List

- ✓ Introduction
- ✓ Previous Work
- ✓ Motivation
- ✓ Calibrated Linear Recursive Structure (CLRS)
- ✓ Results
- ❑ **Conclusion & Future Work**

Conclusion

- Learned CDF, when used as a hash function, has the potential to improve bucket utilization, collision rate and average bucket height
- The improvements don't hold for all distributions: when the CDF is very non-linear, the experts may have difficulty learning piecewise distributions
- Besides usage pattern and architecture of the data structure, how much benefit we can reap from learned indexes is also influenced by the underlying data distribution

Future Work

- Additional model complexity
- Adaptive expert assignment: more experts for ranges in CDF that are hard to learn
- Principles for designing hybrid architecture:
    - e.g. when memory is limited, how much should we improve stage 1 model vs increase number of experts?
- Multidimensional data (joint distribution)

Bucket List

- ✓ Introduction
- ✓ Previous Work
- ✓ Motivation
- ✓ Calibrated Linear Recursive Structure (CLRS)
- ✓ Results
- ✓ Conclusion & Future Work

Thank You

References:

1. Gupta, Maya, et al. "Monotonic calibrated interpolated look-up tables." *The Journal of Machine Learning Research* 17.1 (2016): 3790-3836.
2. Howard, Andrew, and Tony Jebara. "Learning monotonic transformations for classification." *Advances in Neural Information Processing Systems*. 2008.
3. Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in Neural Information Processing Systems*. 2017.
4. Kraska, Tim, et al. "The Case for Learned Index Structures." *Proceedings of the 2018 International Conference on Management of Data*. ACM, 2018.