智能数据库技术：理论到实践

丁博麟

阿里巴巴达摩院-智能计算实验室-智能系统

智能系统团队研究方向

Lab members and Research interns

数据隐私

- 隐私法规监管和隐私技术
- 从算法技术、系统、到应用
- 隐私机制、市场效率、和社会伦理

智能系统

- AI4Database: 智能助力数据系统
- AI4AI: 智能助力机器学习系统
- AI4Econ: 智能助力营销策略

Where DB Meets ML

- Human involved in research/engineering/analyzing/administrating:
  - Building and maintaining indexes
  - Building histograms
  - Query optimization
  - Physical design tuning
  - Optimizing view materialization

- Learning to automatically designing/optimizing/tuning?

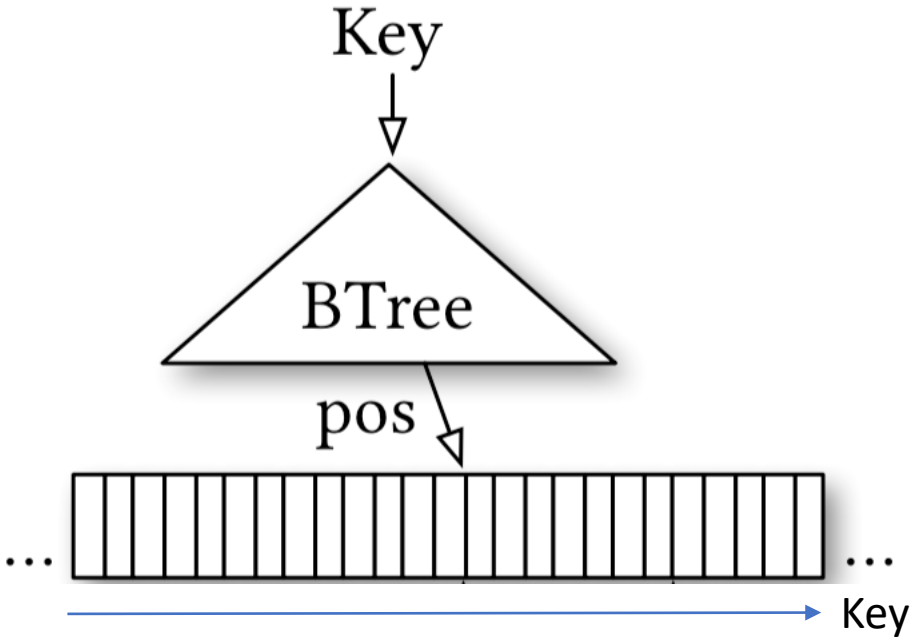
Outline

- Learning to index [a brief survey and analysis]
- Learning to be a statistician: NDV estimation [VLDB2022]
- Learning to optimize: a case for AQP [SIGMOD2022]
- Deployment: Baihe (百合) framework

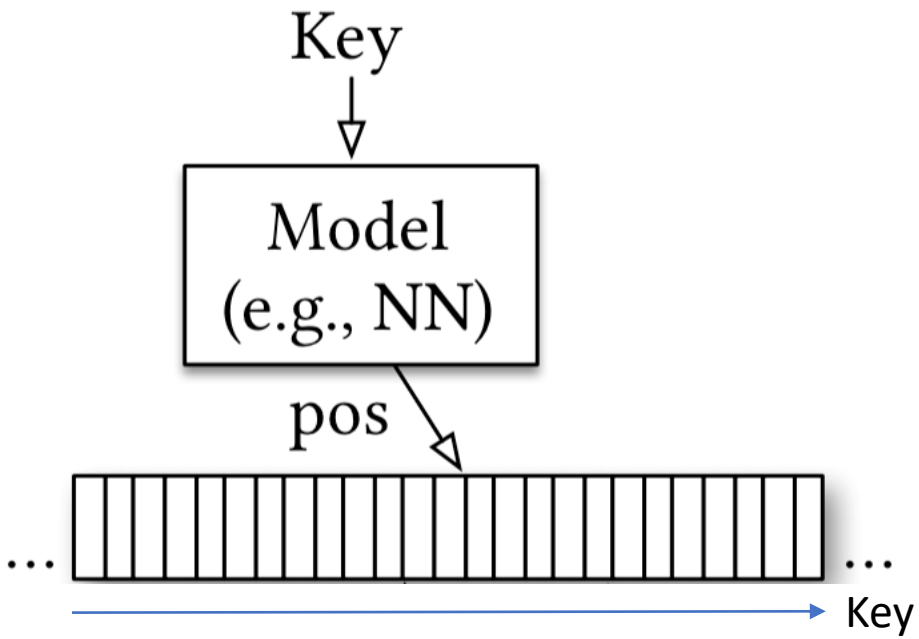
Outline

- Learning to index [a brief survey and analysis]
- Learning to be a statistician: NDV estimation [VLDB2022]
- Learning to optimize: a case for AQP [SIGMOD2022]
- Deployment: Baihe (百合) framework

B-Tree Index from Learning Perspective



Input: Key
Output: Position
B-Tree Index: position = B-tree(Key)



Input: Key
Output: Position
Learned Index: position = function(Key)

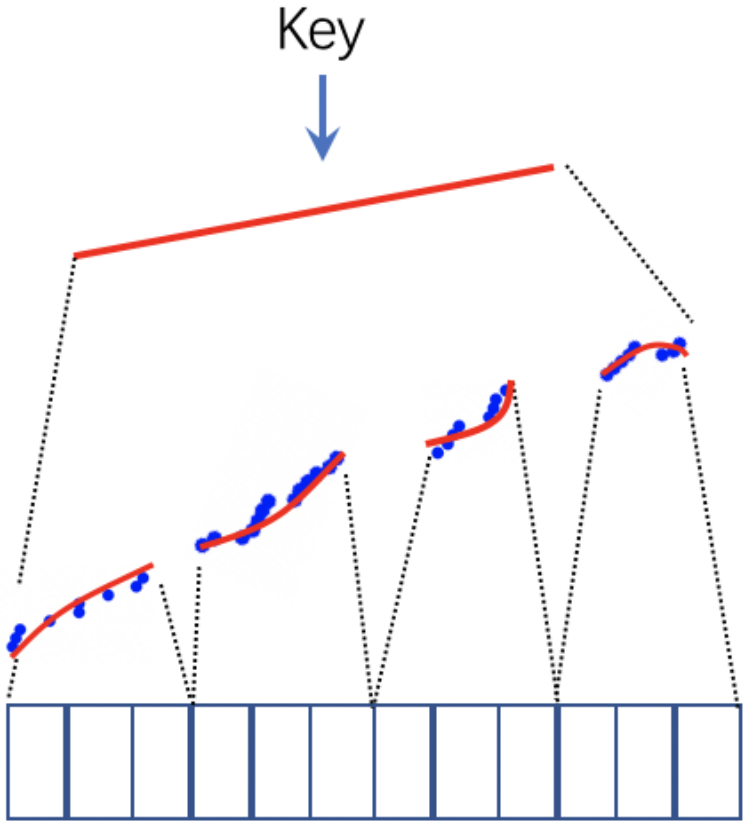
[Image source] Kraska et al., The case for learned index structures. *SIGMOD*, 2018

Why **Learning** Index from Data?

- Consider this (ideal) case: build an index to store and query over a table of *n* rows with continuous integer keys, *i.e.*, Keys = [11, 12, 13, 14, 15, ...] and Pos = [0, 1, 2, 3, 4, …]
  - B-Tree: seeking Pos in time O(log *n*)
  - a learned function Pos = M(Key) = Key + *offset* : O(1)

- Main motivation: the **hidden yet useful distribution information** about the data to be indexed has not been fully explored and utilized in the classic index techniques
  - learned index: an automatic way to explore and utilize such information

Recursive-Model Index (RMI)



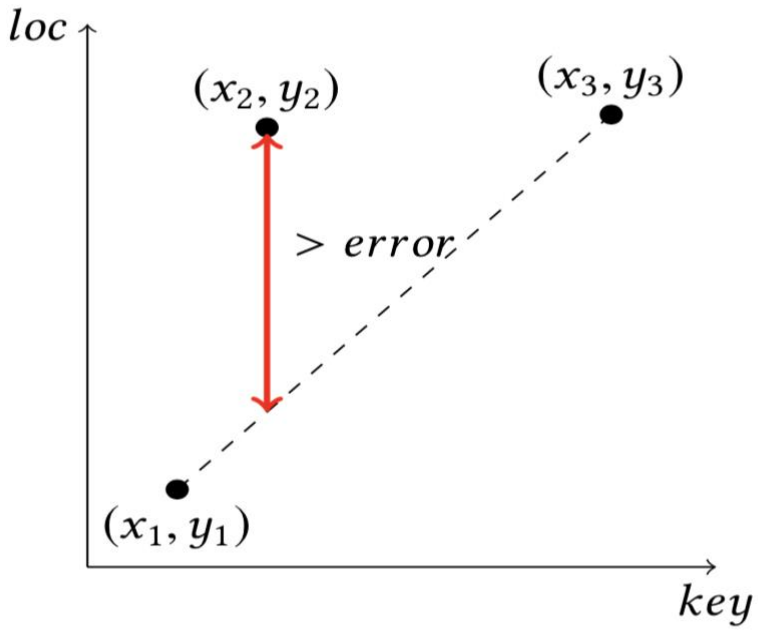
Root model

Sub-models

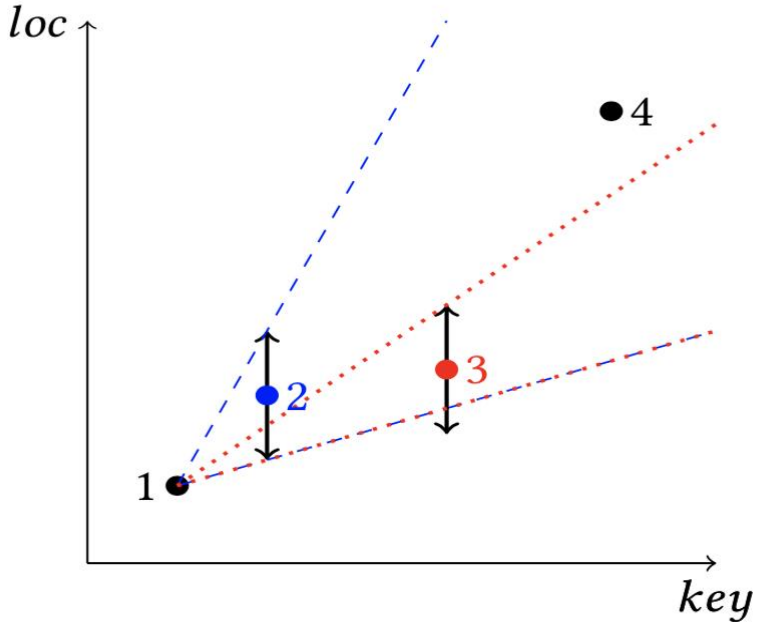
Data to be indexed

[Image source] Kraska et al., The case for learned index structures. *SIGMOD*, 2018

FITing-Tree



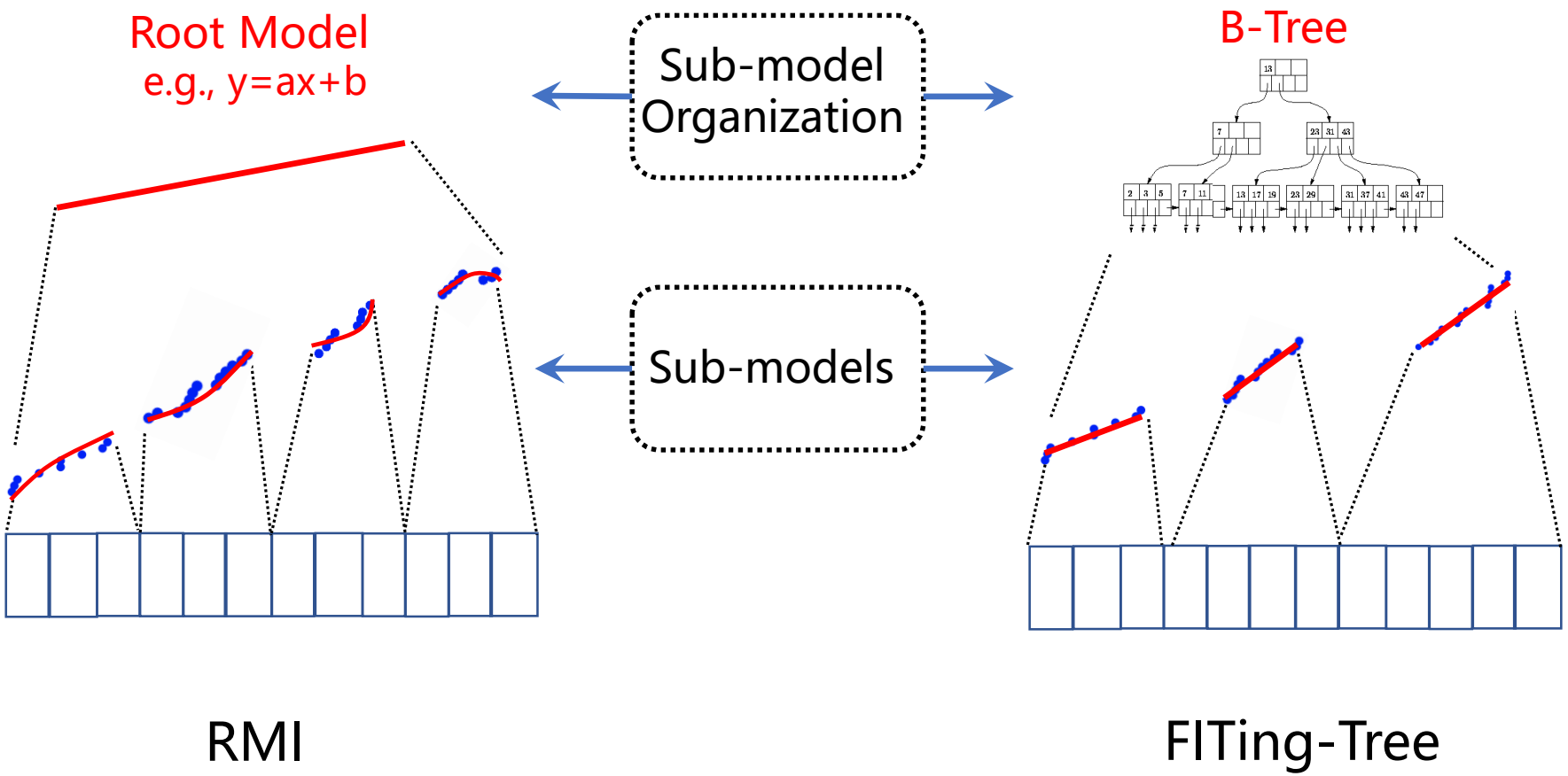
Error-Bounded Linear Segment: Given threshold $error$, a segment from (x_1, y_1) to (x_3, y_3) is *not valid* if (x_2, y_2) is further than $error$ from the interpolated line.



ShrinkingCone (building a segment): Point 1 is the origin of the cone. Point 2 is then added, resulting in the dashed cone. Point 3 is added next, yielding in the dotted cone. Point 4 is outside the dotted cone and therefore starts a new segment.

[Image source] Galakatos et al., FITing-Tree: A Data-aware Index Structure. SIGMOD, 2019

RMI v.s. FITing-Tree



More Learned Index Methods

- PGM [1] improves FITing-Tree by finding the optimal number of learned segments given an error bound.
- ALEX [2] proposes an adaptive RMI with workload-specific optimization, achieving high performance on dynamic workloads.
- RadixSpline [3] gains competitive performance with a radix structure while using a single-pass training.
- Multi-dimensional indexes: NEIST [4], Flood [5], Tsunami [6] and LISA [7].

More Learned Index Methods

[1] The PGM-Index: A Fully-Dynamic Compressed Learned Index with Provable Worst-Case Bounds. *PVLDB*, 2020.

[2] ALEX: An Updatable Adaptive Learned Index. *SIGMOD*, 2020.

[3] RadixSpline: A Single-Pass Learned Index. *In aiDM Workshop on SIGMOD*, 2020.

[4] NEIST: a Neural-Enhanced Index for Spatio-Temporal Queries. *TKDE*, 2019.

[5] Learning Multi-dimensional Indexes. *SIGMOD*, 2020.

[6] Tsunami: A Learned Multi-dimensional Index for Correlated Data and Skewed Workloads. *PVLDB*, 2020.

[7] LISA: A Learned Index Structure for Spatial Data. *SIGMOD*, 2020.

Questions about Learned Indexes

How to systematically analyze and design machine learning based indexing methods?

More scalable index learning methods?

Which class of models suffice?

Task Definition

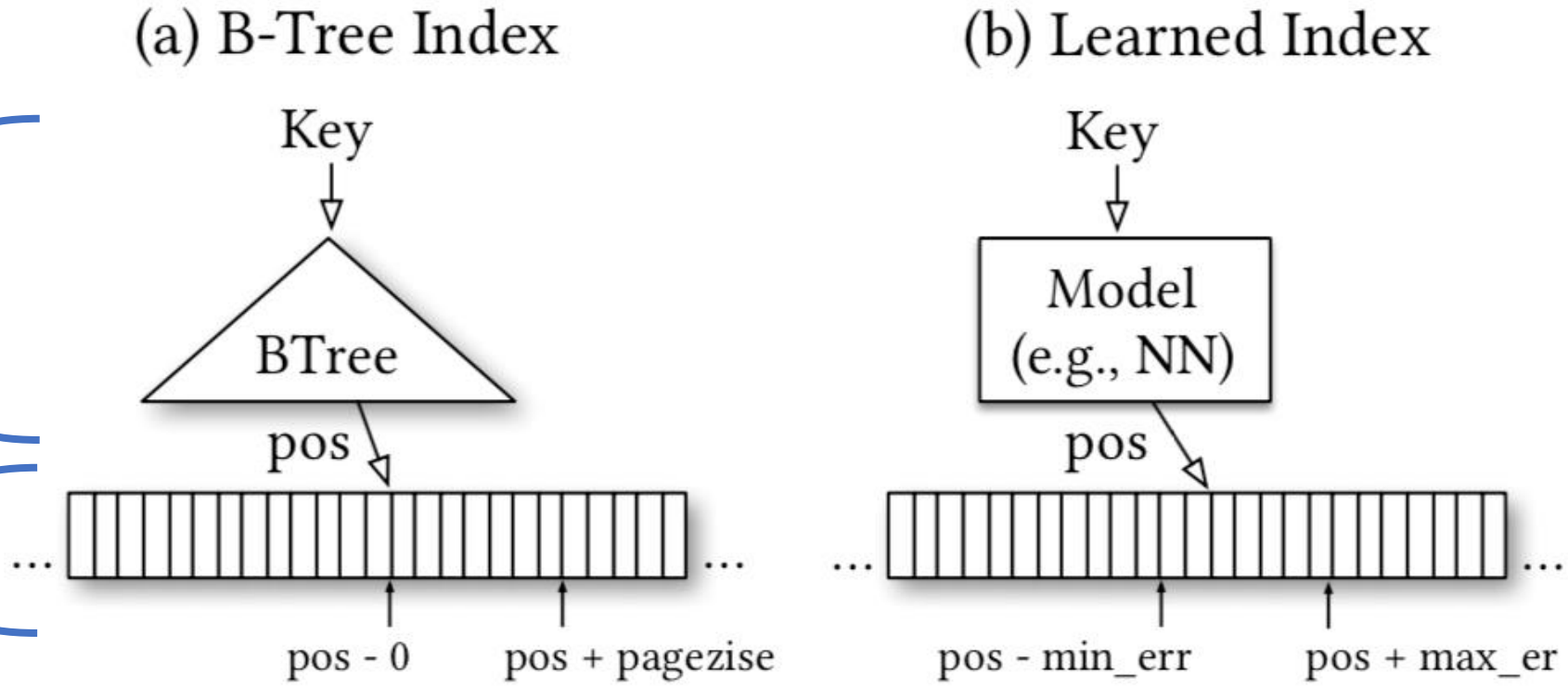
- Given a database $\mathcal{D}$ with n records (rows), let's assume that a *range index* structure will be built on a specific column x. For each record $i \in [n]$, the value of this column, x_i , is adopted as the *key*, and y_i is the *position* where the record is stored.

- We want to learn a *mechanism* M with the key x as input and outputs a *predicated position* $\hat{y} \leftarrow M(x)$ for accessing data.

Learning Index: A Machine Learning Task

$L(M)$
measures the cost of calculating $\hat{y} \leftarrow M(x)$

$L(\mathcal{D}|M)$
$= \mathbb{E}_{\{x,y\}\in\mathcal{D}} L(y, \hat{y})$



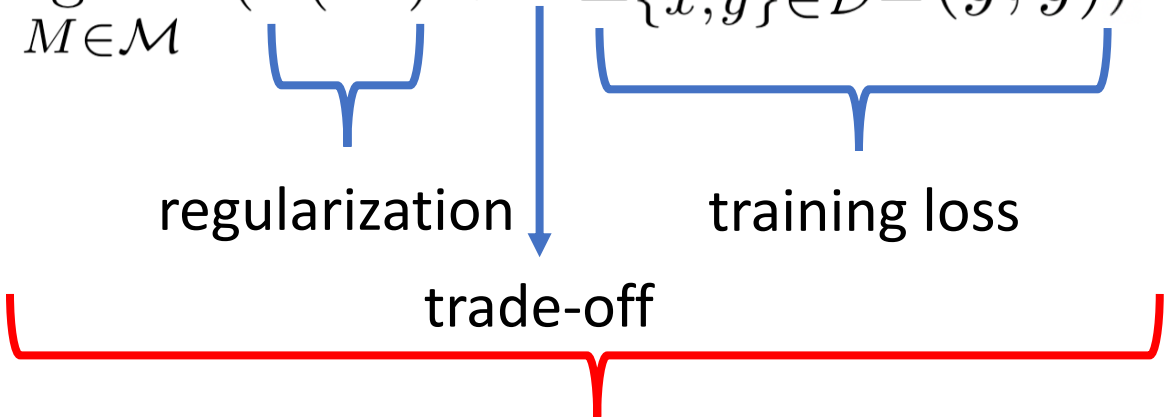
A Pluggable Learned Index Method via Sampling and Gap Insertion, https://arxiv.org/pdf/2101.00808.pdf

Learning Index: A Machine Learning Task

$$M^* = \arg\min_{M \in \mathcal{M}} \mathrm{MDL}(M, \mathcal{D})$$

$$= \arg\min_{M \in \mathcal{M}} \left(L(M) + \alpha L(\mathcal{D}|M)\right)$$

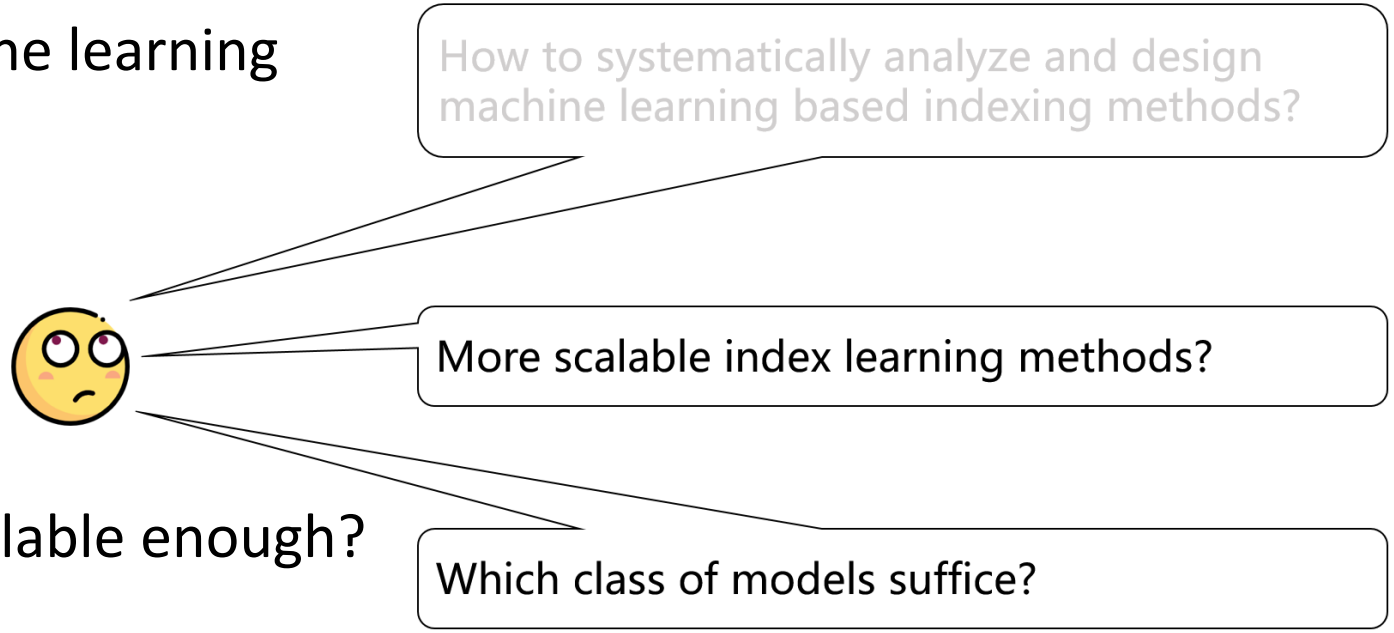
$$= \arg\min_{M \in \mathcal{M}} \left(L(M) + \alpha\, \mathbb{E}_{\{x,y\} \in \mathcal{D}} L(y, \hat{y})\right)$$



A Pluggable Learned Index Method via Sampling and Gap Insertion, https://arxiv.org/pdf/2101.00808.pdf

Benefits of Learned Index

- Smaller Size
- Faster Index Seek
- Better Handling Index Update (*hopefully*)
  - Generalization ability of machine learning
  - Incremental learning
- Question Mark
  - Is model training/inference scalable enough?



Learned Index with Sampling

$$M^* = \arg\min_{M \in \mathcal{M}} \mathrm{MDL}(M, \mathcal{D})$$
$$= \arg\min_{M \in \mathcal{M}} (L(M) + \alpha L(\mathcal{D}|M))$$
$$= \arg\min_{M \in \mathcal{M}} (L(M) + \alpha \, \mathbb{E}_{\{x,y\} \in \mathcal{D}} L(y, \hat{y}))$$

regularization | training loss | trade-off | objective function

- How large the sample needs to be?
  - n is the data size
  - M^* is fully optimized

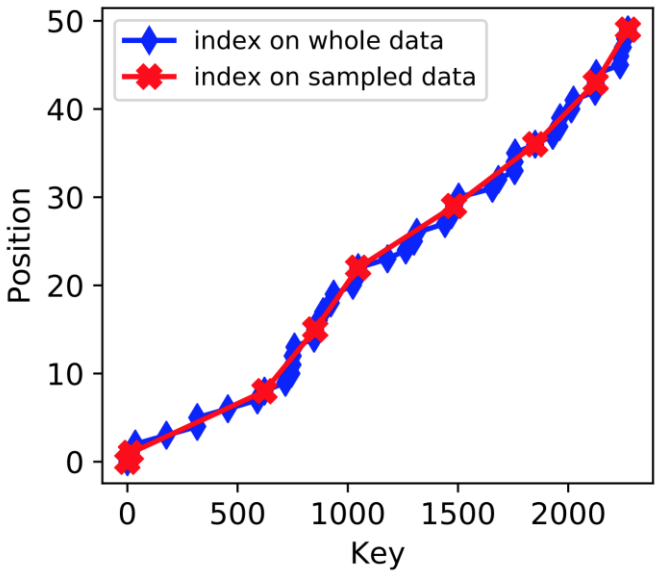


Fig: Illustration of sampling

THEOREM 1. *Consider the optimization problem:*

$$M^* = \arg\min_{M \in \mathcal{M}} \mathrm{MDL}(M, \mathcal{D}) = \arg\min_{M \in \mathcal{M}} (L(M) + \alpha L(\mathcal{D}|M)) .$$

We can solve it on a random sample D_s *with size* $s = \mathrm{O}(\alpha^2 \log^2 n)$ *as*

$$\hat{M}^* = \arg\min_{M \in \mathcal{M}} \mathrm{MDL}(M, \mathcal{D}_s)$$

s.t., $\mathrm{MDL}(\hat{M}^*, \mathcal{D}) \leq \mathrm{MDL}(M^*, \mathcal{D}) + \mathrm{O}(1)$ *with high probability.*

A Pluggable Learned Index Method via Sampling and Gap Insertion, https://arxiv.org/pdf/2101.00808.pdf

Learned Index with Sampling

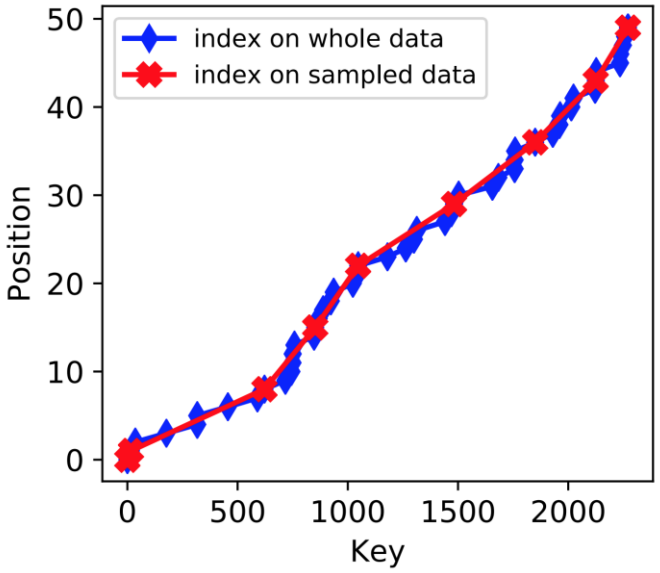
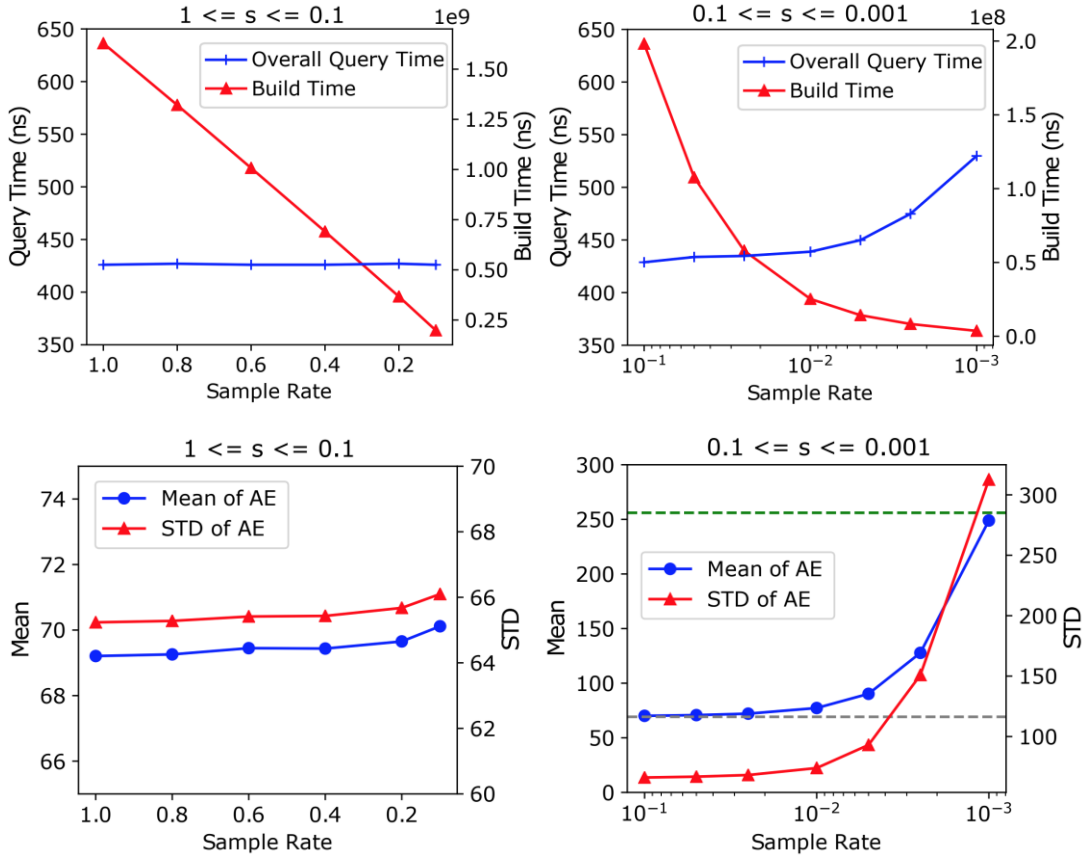


Fig: Illustration of sampling

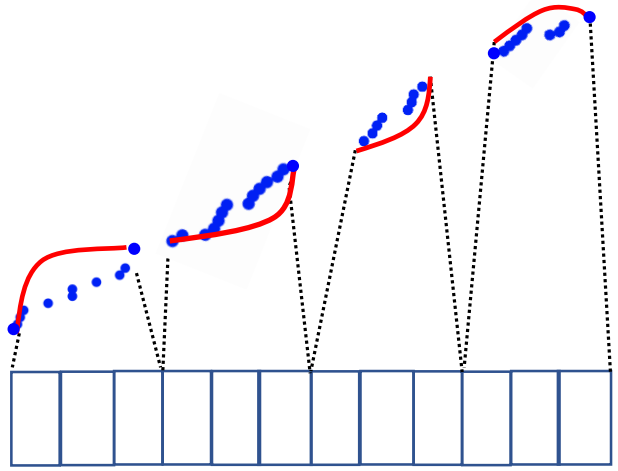
- Up to **78x** building speedup
- **Non-degraded performance** in terms of query time and prediction error)



A Pluggable Learned Index Method via Sampling and Gap Insertion, https://arxiv.org/pdf/2101.00808.pdf

Is Linear Model Sufficient?

- Linearization of a learned model

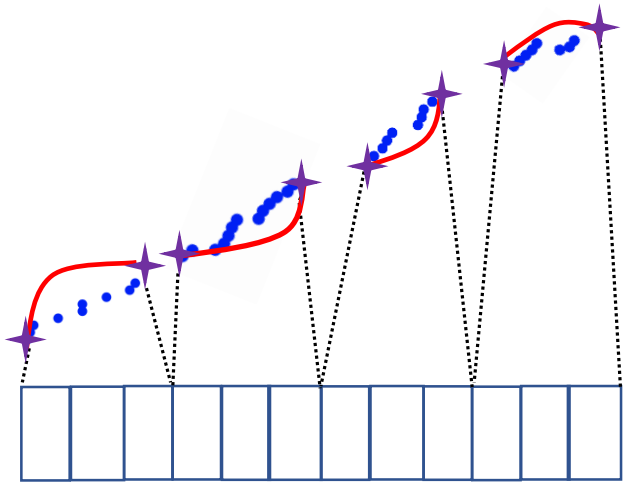


A learned model

$$\bar{y} = M(x)$$

Is Linear Model Sufficient?

- Linearization of a learned model

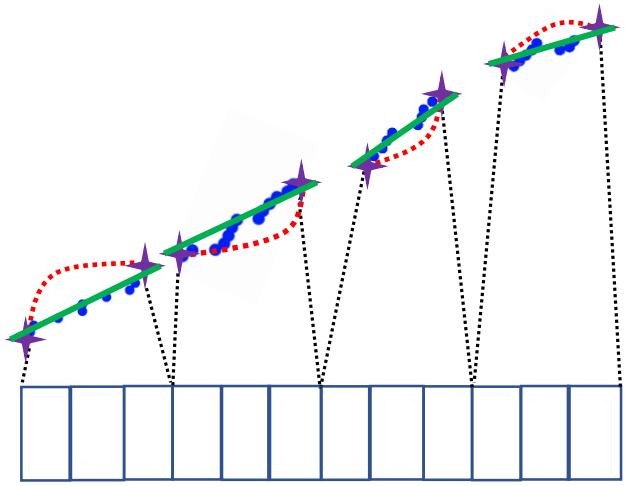


A learned model $\bar{y} = M(x)$

Landmark points $\ldots, (x_l, y_l), (x_r, y_r), \ldots$

Is Linear Model Sufficient?

- Linearization of a learned model



A learned model $\bar{y} = M(x)$

Landmark points $\ldots, (x_l, y_l), (x_r, y_r), \ldots$

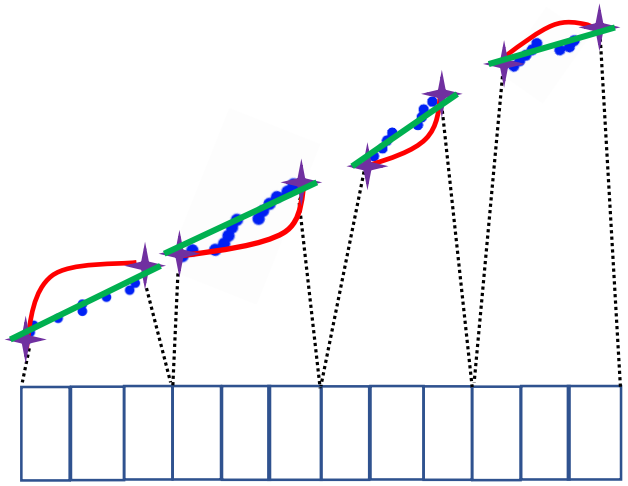
Linearized model $\hat{y} = M_L(x)$

connecting $(x_l, \bar{y}_l = M(x_l))$ **to** $(x_r, \bar{y}_r = M(x_r))$

Is Linear Model Sufficient?

Yes! As long as landmark points are dense enough

- Linearization of a learned model



A learned model $\bar{y} = M(x)$

Landmark points $\ldots, (x_l, y_l), (x_r, y_r), \ldots$

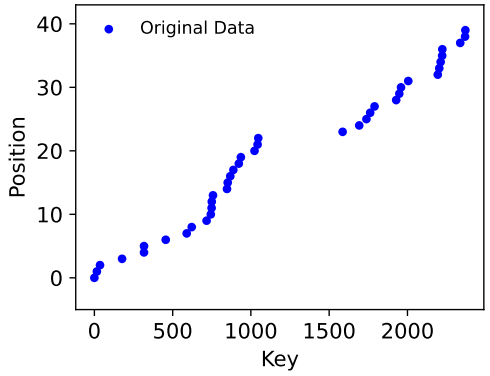
Linearized model $\hat{y} = M_L(x)$

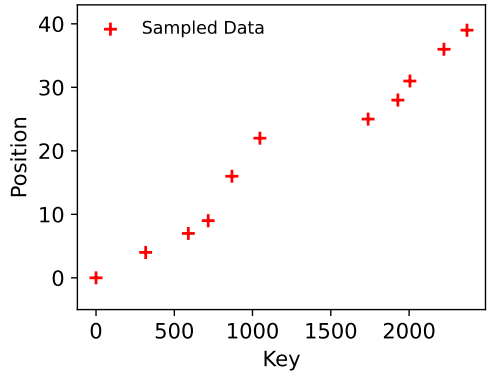
connecting $(x_l, \bar{y}_l = M(x_l))$ **to** $(x_r, \bar{y}_r = M(x_r))$

Theorem 2. Suppose $\forall x, |\bar{y} - y| \leq \epsilon$, after linearization, we have $\forall x, |\hat{y} - y| \leq 3\epsilon + 2(y_r - y_l)$.

A Pluggable Learned Index Method via Sampling and Gap Insertion, https://arxiv.org/pdf/2101.00808.pdf

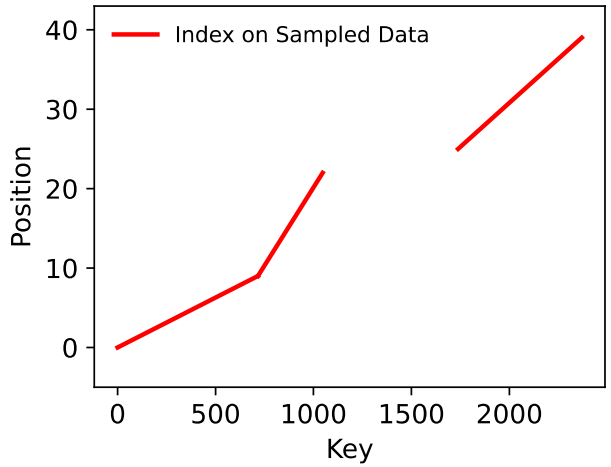
Sampling-Restriction-Linearization

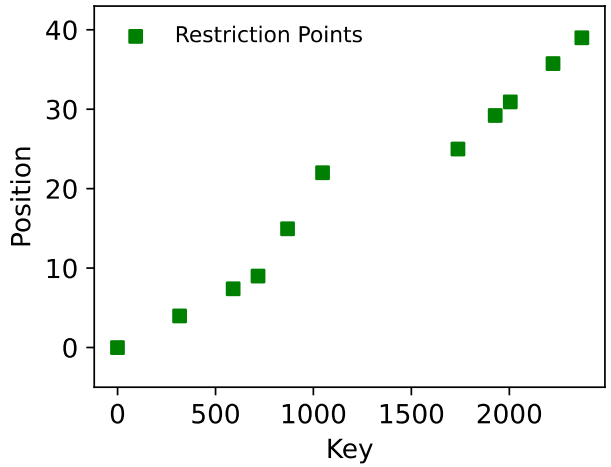


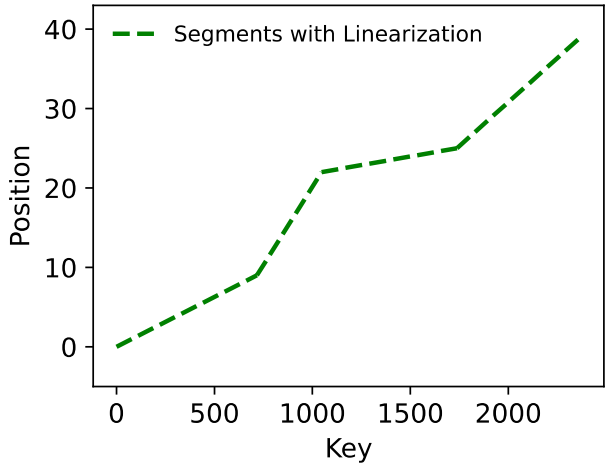


Sampled data points as landmark points:
$\ldots, (x_l, y_l), (x_r, y_r), \ldots$

Theorem 2. Suppose $\forall x, |\overline{y} - y| \leq \epsilon$, after linearization, we have $\forall x, |\hat{y} - y| \leq 3\epsilon + 2(y_r - y_l)$.







Learned Index on Sampled Data → **Restriction** → **Linearization**

bounding ϵ | bounding $y_r - y_l$ | bounding $|\hat{y} - y|$

A Pluggable Learned Index Method via Sampling and Gap Insertion, https://arxiv.org/pdf/2101.00808.pdf

Still Open…

- How to build it into real DB systems?
  - Without too much modification to the current system
  - Index seek v.s. the whole plan: worth a try?

Outline

- Learning to index [a brief survey and analysis]
- Learning to be a statistician: NDV estimation [VLDB2022]
- Learning to optimize: a case for AQP [SIGMOD2022]
- Deployment: Baihe (百合) framework

Learning to be a statistician

Statistician's task: data $\mathbf{C} \rightarrow$ some statistics, e.g., $\mathrm{mean}(\mathbf{C})$, $\mathrm{median}(\mathbf{C})$, $\mathrm{NDV}(\mathbf{C})$, …

Estimating statistics from random sample:

- Draw a sample $\mathbf{S}$ from $\mathbf{C}$
- Derive some formula (estimator) $\mathrm{h}(\mathbf{S}) \approx \mathrm{stat}(\mathbf{C})$
- **Desirable property:** unbiasedness, small variance (or sampling efficiency), consistency, …

Can we learn h**?**

Input feature: $\mathbf{S}$
(drawn from $\mathbf{C}$)

Alchemy: h
(train a machine learning model to approximate what a statistician derives)

$\mathrm{h}(\mathbf{S}) \approx \mathrm{stat}(\mathbf{C})$

Technical challenges:

- How to generate training data?
- What are the features?
- What model/loss function should be used?
- Workload-dependent? (able to be generalized?)

Learning to be a statistician: a toy example

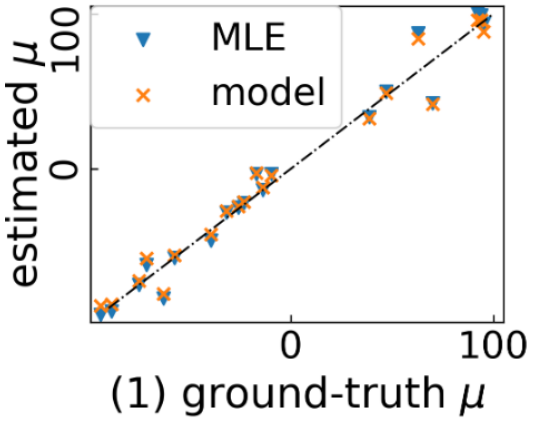
Consider a Gaussian distribution $N(\mu, \sigma^2)$, and a sample $\mathbf{S} = \{v_1, v_2, \dots, v_k\}$
How to estimate μ and σ?

Statistician: $\mu^{\text{MLE}} = \frac{1}{k}\sum v_i$; $\sigma^{\text{MLE}} = \sqrt{\frac{1}{k}\sum (v_i - \mu^{\text{MLE}})^2}$

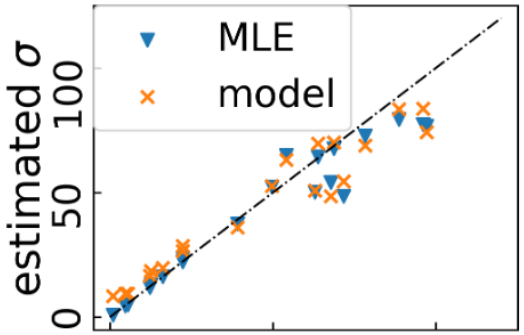
Both $\hat{\mu}$ and $\hat{\sigma}$ are in the form $\text{h}(\mathbf{S})$… Can we learn it from samples?

Prepare training data points and feed into an NN model:

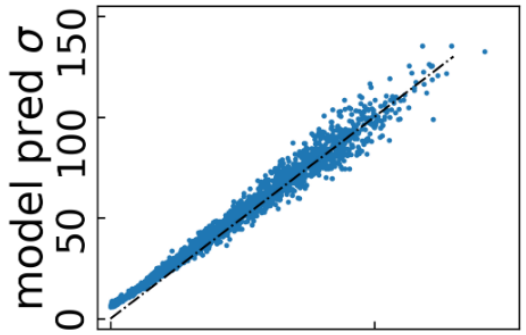
Features (size-4 sample => 4-dim features)	Label μ	Label σ
(from $N(0,100)$) => {-1.2, -3.6, 3.2, 2.1}	0	100
(from $N(2,80)$) => {1.1, 2.2, 3.4, 0.3}	2	80
(from $N(6,200)$) => {0, 4.2, 6.3, 18.7}	6	200
(from $N(-2,10)$) => {-1.6, -1.9, -4, 0.2}	-2	10



(1) ground-truth μ



(2) ground-truth σ



(3) MLE of σ

Estimating number of distinct values (NDV) from random sample

A value $i \in \Omega$
Data column C: a multi-set of N values
A random sample S $\subseteq$ **C**: n values drawn uniformly at random from **C**

$$\mathbf{C} = \{1, 1, 1, 1, 2, 2, 3, 3, 4, 5, 6, 7\}$$
$$\mathbf{S} = \{1, 1, 2, 3, 4, 7\}$$
$$D = 7$$
$$d = 5$$

Sampling rate $\alpha = n/N$
NDV D and **sample NDV** d

Frequency (N_i): # times i appears in **C**
Profile (F_j): # distinct values with frequency j in **C**
Deterministic relation: $D = \sum_{j>0} F_j$

$$(N_i) = (4, 2, 2, 1, 1, 1, 1)$$
$$(F_i) = (4, 2, 0, 1, \dots)$$

Sample frequency (n_i): # times i appears in **S**
Sample profile (f_j): # distinct values with sample frequency j in **S**

$$(n_i) = (2, 1, 1, 1, 0, 0, 1)$$
$$(f_i) = (4, 1, 0, 0, \dots)$$

A MLE-based formulation

MLE: maximum likelihood estimation

Probability distribution of sample profile f: $\mathbb{P}(f \mid F)$

Feasible D-NDV profile configurations: $\mathcal{F}(D) = \{F \mid \sum_{j>0} F_j = D\}$

We can estimation:

$$D^{\mathsf{MLE}} = \arg\max_{D} \sum_{F \in \mathcal{F}(D)} \mathbb{P}(f \mid F)$$

Easy to be formulated but difficult to be solved ;(

Existing estimators

Shlosser [Shlosser 1981]: works well on skewed datasets with $\frac{\mathrm{E}[f_i]}{\mathrm{E}[f_1]} \approx \frac{F_i}{F_1}$

$$D^{\text{Shlosser}} = d + \frac{f_1 \cdot \sum_{i=1}^{n} (1-\alpha)^i \cdot f_i}{\sum_{i=1}^{n} i \cdot \alpha(1-\alpha)^{i-1} \cdot f_i}$$

Chao [Chao 1984]: approximate $\mathrm{E}[D]$ with N approaches infinity

$$D^{\text{Chao}} = d + \frac{f_1^2}{2f_2}$$

GEE [Charikar, Chaudhuri, Motwani, Narasayya 2000]: a "worst-case" optimal estimator

$$D^{\text{GEE}} = \sqrt{\frac{1}{\alpha}} \cdot f_1 + \sum_{i=2}^{n} f_i$$

Negative results [Charikar, Chaudhuri, Motwani, Narasayya 2000]:

$$\text{error}(\hat{D}) = \max\{\hat{D}/D, D/\hat{D}\}$$

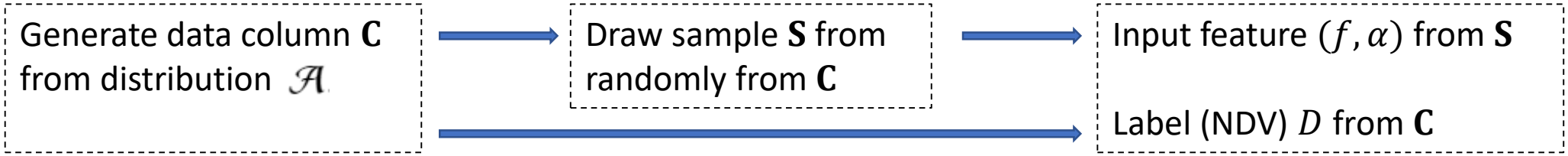
w.h.p.

$$\text{error}(\hat{D}) \geq \Omega(\sqrt{N/n})$$

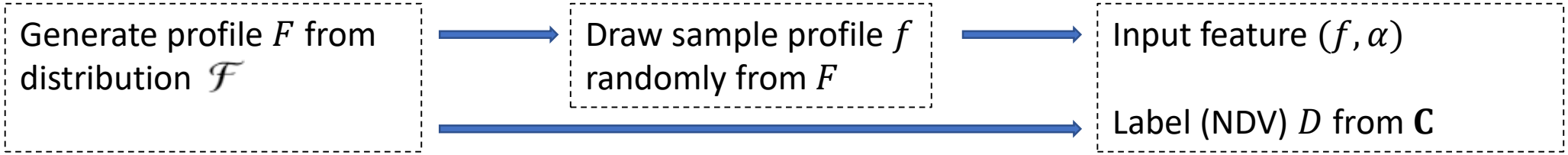
Intuition: hard to distinguish a dataset with NDV = N and one with NDV = n

A (workload-agnostic) learning framework

Generating training sample:



Or equivalently:



Train a machine learning model h **to approximate (recall the MLE formulation**

$$D^{\mathsf{MLE}} = \arg\max_{D} \sum_{F \in \mathcal{F}(D)} \mathbb{P}(f \mid F)$$

$$\hat{D} = \mathrm{h}(f, \alpha) \approx \arg\max_{D} \mathbb{P}_{\mathcal{F}}(f \mid D)$$

Ratio Error: $\mathrm{error}(\hat{D}) = \max\{\hat{D}/D, D/\hat{D}\}$

Loss function and empirical loss:

$$L(\mathrm{h}(f, \alpha) = \hat{D}, D) = |\log \hat{D} - \log D|^2 = \left(\log \mathrm{error}(\hat{D})\right)^2$$

$$\hat{R}_A(h) = \sum_{((f,\alpha),D) \in A} L(\mathrm{h}(f, \alpha), D)$$

How to use the learned estimator

- Load the learned estimator (“model_paras.npy”)
- Generate profile from the sample (sample: pre-drawn/block-level/top-k)
- Use profile as the input feature to predict NDV

(implemented in Alibaba MaxCompute-ODPS)

https://figshare.com/s/0415ccdd6d19645e44b5

```
class Estimator:
    def __init__(self, para_path="model_paras.npy"):
        self.paras = np.load(para_path, allow_pickle=True)

    def sample2profile(self,S):
        _, counts = np.unique(S, return_counts=True)
        cnt, val = np.unique(counts, return_counts=True)
        fs = np.zeros(np.max(cnt)).astype(int)
        fs[cnt - 1] = val
        return fs

    def sample_predict(self,S,N):
        """
        estimate number of distinct values in population from a sample S
        :param S: a sample, list
        :param N: population size, int
        :return: estimated number of distinct values, int
        """
        f = self.sample2profile(S)
        f_sparse = to_sparse(f)
        ndv = self.profile_predict([f_sparse],[N],is_sparse=True)
        return ndv
```

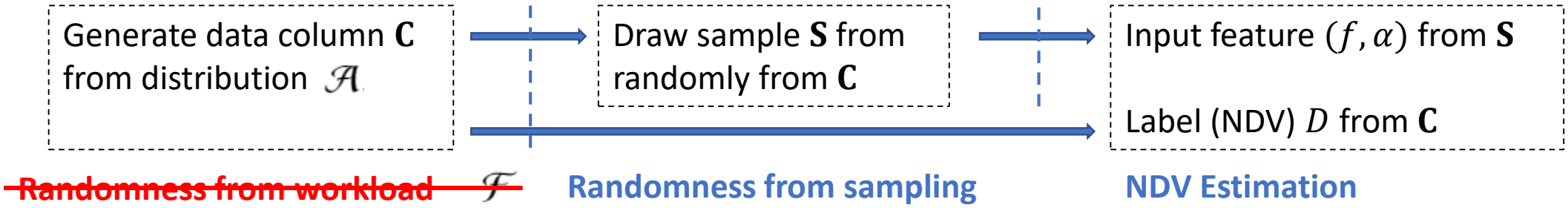
https://www.consultantsmind.com/2016/07/20/are-you-sure/

- Synthetic training data v.s. estimation on real workload?
- Proper feature?
- Hardness result?

How to make the model workload-agnostic?

$$\hat{D} = \mathrm{h}(f, \alpha) \approx \arg\max_{D} \mathbb{P}_{\mathcal{F}}(f \mid D)$$

What to be learned?



$$\mathbb{P}_{\mathcal{F}}(f \mid D) = \sum_{F \in \mathcal{F}(D)} \mathbb{P}(f, F \mid D)$$

$$= \sum_{F \in \mathcal{F}(D)} \mathbb{P}(f \mid F, D) \cdot \mathbb{P}(F \mid D) = \sum_{F \in \mathcal{F}(D)} \mathbb{P}(f \mid F, D) \cdot \frac{\mathbb{P}(F, D)}{\mathbb{P}(D)}$$

$$= \sum_{F \in \mathcal{F}(D)} \mathbb{P}(f \mid F) \cdot \frac{\mathbb{P}(F)}{\mathbb{P}(D)}$$

Choose the distribution $\mathcal{F}$ carefully

If $\mathbb{P}(F)/\mathbb{P}(D)$ is a constant:

$$\mathbb{P}_{\mathcal{F}}(f \mid D) \sim \sum_{F \in \mathcal{F}(D)} \mathbb{P}(f \mid F)$$

approximation to an MLE estimator

$$\arg\max_{D} \mathbb{P}_{\mathcal{F}}(f \mid D) = \arg\max_{D} \sum_{F \in \mathcal{F}(D)} \mathbb{P}(f \mid F) = D^{\mathrm{MLE}}$$

Rest questions

- **Feature reduction/selection**
- **Training sample generation**
- **Model structure**
- **Hardness results v.s. Model regularization**

Feature reduction

- **From sample to profile**: profile suffices for "consistent" estimator (proved in the paper)
- **Dimensionality reduction** $\{f_1, f_2, \dots, f_n\} \to \{f_1, f_2, \dots, f_B\}$: from n-dim to B-dim, **why?**

Frequent (Sample-frequent) value i: $N_i > B$ $(n_i > B)$

a) **Infrequent -> sample-infrequent**: stay in the sample, with probability roughly α
b) **Infrequent -> disappearing**: unobserved, with probability roughly $1 - \alpha$
c) **Frequent -> sample-frequent**: stand out
d) **Frequent -> sample-infrequent**: mixed with a)
e) **Frequent -> disappearing**: with low probability (roughly $1/e^{\alpha B}$)

Choose $B \geq (1/\alpha) \ln N$

a) + d)	c)	e)

$$\mathrm{h}(f, \alpha) \approx \mathrm{g}(f_1, f_2, \dots, f_B) + \sum_{j=B+1}^{n} f_j + \mathrm{O}(1)$$

Feature set 1: $\{f_1, f_2, \dots, f_B\}$
Feature 2: $\left\{f_{B+} = \sum_{j>B} f_j\right\}$

Training sample generation

A profile distribution with constant $\mathbb{P}(F)\ /\mathbb{P}(D)$
(F follows uniform distribution, conditioned on D)

Draw $F \sim \mathcal{F}$ uniformly at random, with constraints

$$N = \sum_i iF_i \ ; \quad D = \sum_i F_i$$

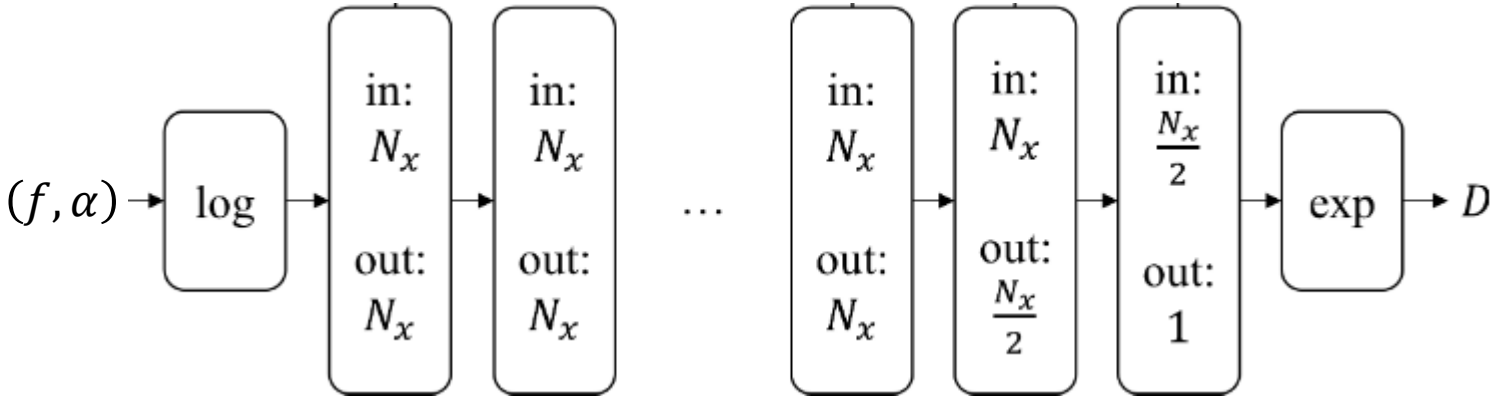
Reduce to *random fixed sum problem*:

$$Q(N, M) = \{SF \mid \sum_{i=1}^{M} SF_i = N \ \ \& \ \ SF_i \geq SF_{i+1} \ \forall \ i\}$$

One-to-one mapping: $F_i = SF_i - SF_{i+1}$
(F_i) satisfies constraints and is uniformly at random

Model structure

Logarithmic -> K + 2 fully connected layers -> Exponential



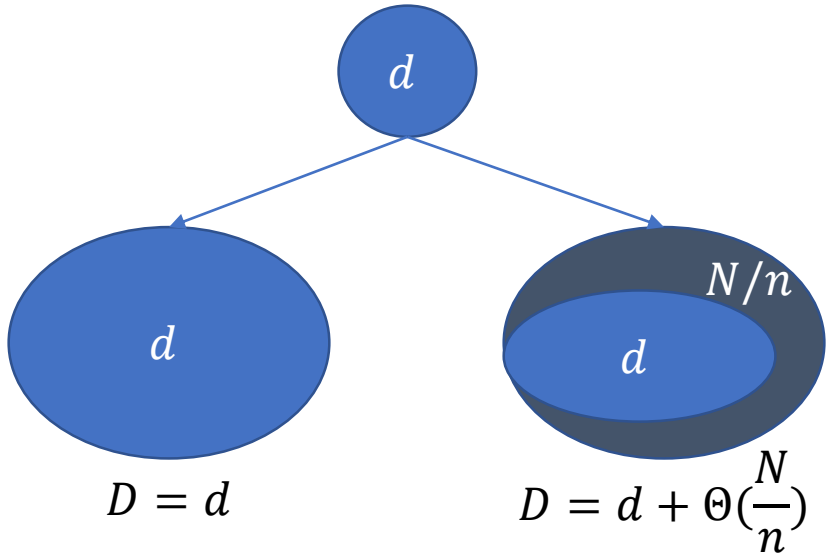
Hardness results v.s. Model regularization

An instance-level hardness result:
Observing a sample with sample NDV d
Hard to distinguish two scenarios (w.h.p.):

1. $D = d$: no unobserved values in the full column
2. $D = d + \Theta(\frac{N}{n})$: there are $\frac{N}{n}$ values unobserved values which appear only once in the full column

Observed sample

Underlying data



Worst-case optimal estimator
$\widehat{D} = \sqrt{d \cdot (d + N/n)}$

Ratio error $\mathrm{error}(\widehat{D})$ **can be as large as**: $\mathrm{maxerr}(d) = \sqrt{\frac{d+N/n}{d}}$ (w.h.p.)

Hardness results v.s. Model regularization

Model regularization:

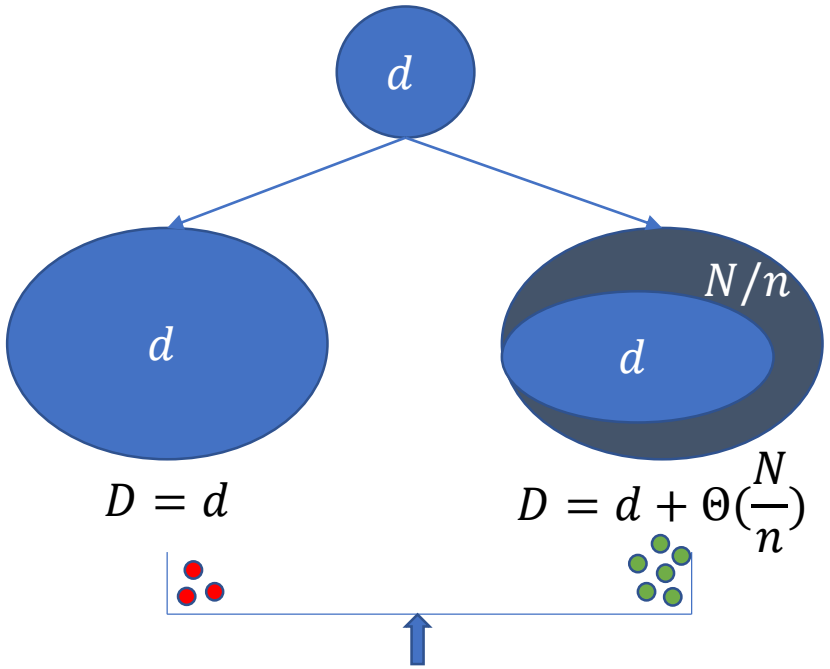
$$L(\mathrm{h}(f,\alpha)=\hat{D},D)=|\log\hat{D}-\log D|^2=\left(\log \mathrm{error}(\hat{D})\right)^2$$

$$L(\mathrm{h}(f,\alpha)=\hat{D},D)=\left|(\log \mathrm{error}(\hat{D}))^2-(\log \mathrm{maxerr}(d))^2\right|$$

Intuition:

Observed sample

Underlying data



Force to learn $\widehat{D}$ in the middle; otherwise adversary may make $\widehat{D}$ lie on the boundary

Experimental evaluation

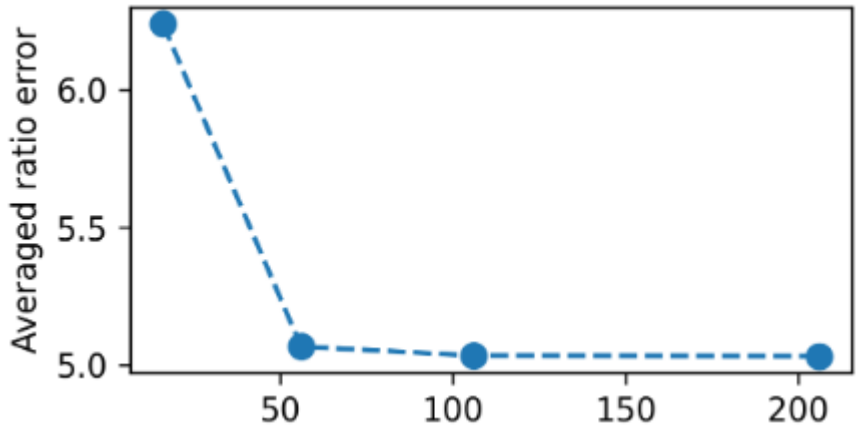
Datasets: 6 public + 3 private
Sampling rate: from 0.0001 to 0.01
Default hyperparameters: $B = 100$, 7 + 2 fully connect layers

Eight previous NDV estimators + **Ours (learned one)** + **LB** (workload-dependent lower-bound)

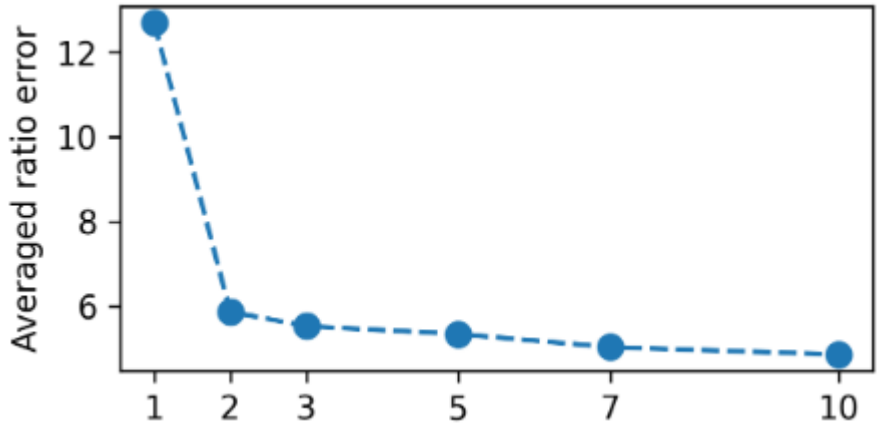
	GEE	HYB GEE	HYB SKEW	AE	Chao	Chao -Lee	Shlo sser	APML	LB	Our
Kasandr	3.4	7.9	8.2	4.6	4.7	9.9	19.2	5.1	2.2	**2.4**
Airline	4.1	1.8	1.8	**1.4**	1.6	1.8	70.0	2.0	2.5	1.9
DMV	5.2	3.5	17.5	7.7	8.8	13.8	23.5	7.6	3.2	**2.7**
Campaign	7.3	6.5	8.4	291.8	13.4	48.0	21.9	13.2	2.9	**3.9**
SSB	5.2	1.2	1.2	**1.1**	**1.1**	1.2	173.1	1.3	1.5	2.0
NCVR	12.2	31.0	56.9	150.2	13.2	58.6	42.6	16.9	5.6	**6.4**
Product	36.3	60.6	58.7	46.1	46.9	250.6	30.5	54.8	9.2	**14.6**
Inventory	17.8	23.5	18.4	75.1	24.8	252.9	11.6	26.7	4.5	**7.8**
Logistics	17.1	93.1	100.0	552.0	15.5	275.1	19.5	16.7	3.6	**3.5**
Average	12.1	25.5	30.1	125.6	14.5	101.3	45.7	16.0	3.9	**5.0**

Experimental evaluation

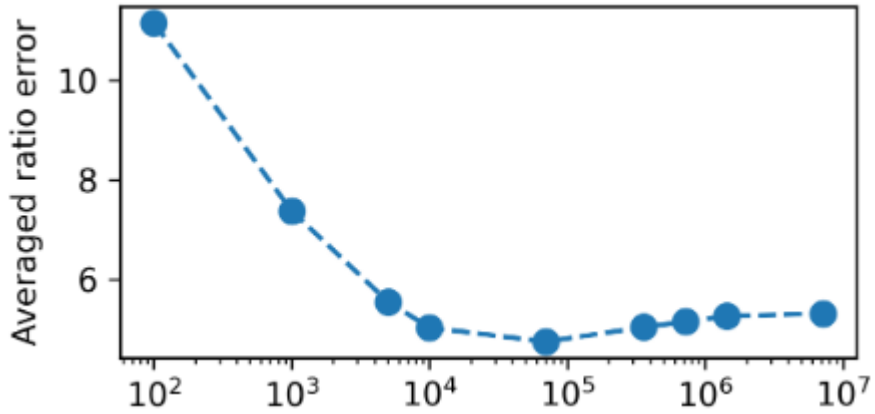
Tuning the model



Feature size B



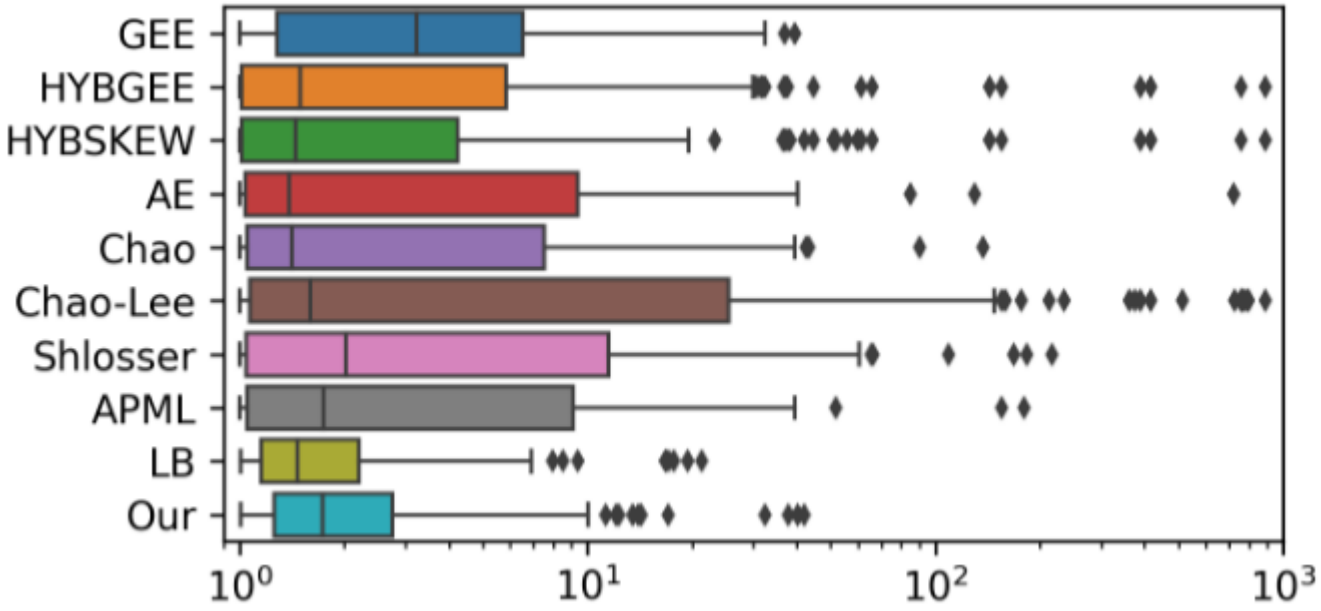
layers in the model



Data size N in training sample

Experimental evaluation

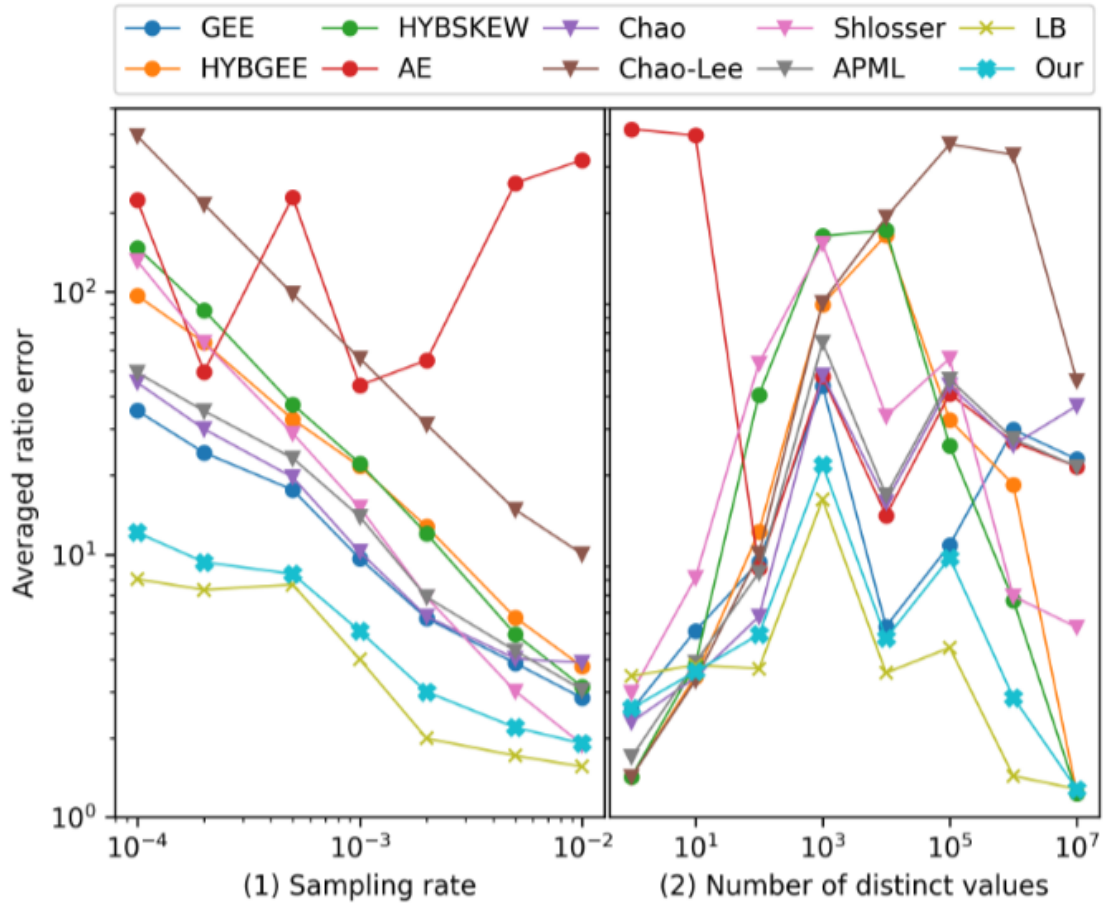
More details



Sampling rate 0.001
Min + 25% + 50% + 75% + Max + Outliers

Experimental evaluation

Average error: different sampling rates and NDVs



Summary

- Attempt to learn an estimator (which is hard to derive)
- Synthetic training sample -> workload-agnostic model/estimator
- Careful feature engineering and model regularization

Future work

- Fine-tuning of the estimator

Pros: even better accuracy

Cons: may not generalize

- Learn other statistics

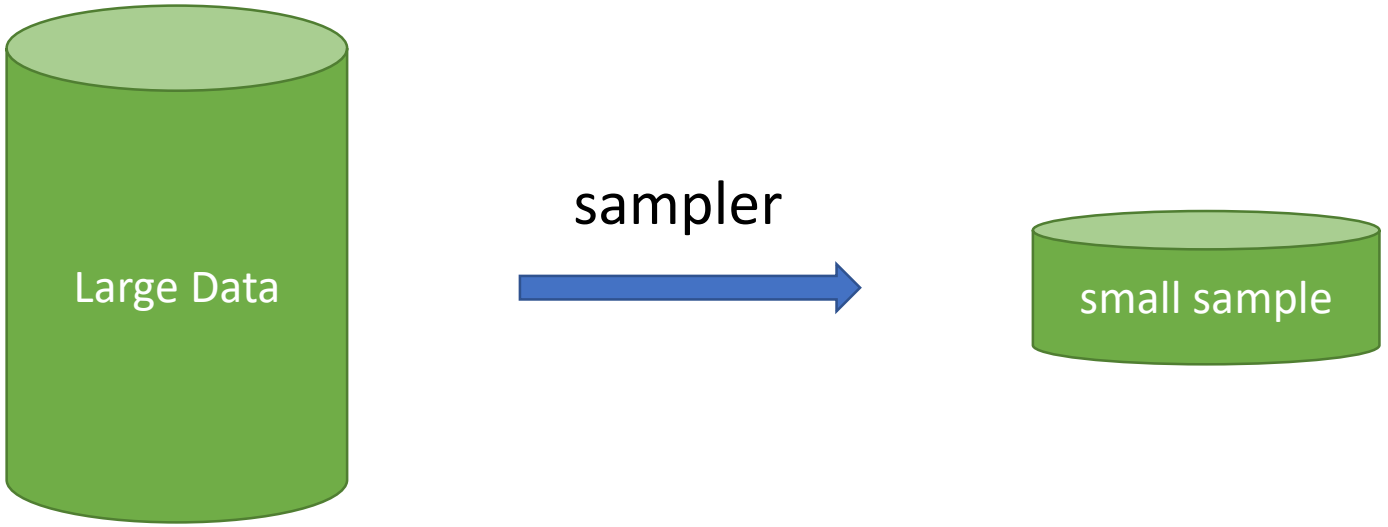
- How it works together with other components, e.g., query optimizer

Outline

- Learning to index [a brief survey and analysis]
- Learning to be a statistician: NDV estimation [VLDB2022]
- Learning to optimize: a case for AQP [SIGMOD2022]
- Deployment: Baihe (百合) framework

Approximate Query Processing (AQP)

SELECT SUM(Price) WHERE Country = ‘CHINA’



AQP: using Sampler

Data

Item	Price	Country
1	0	CHINA
2	50	CA
3	300	CHINA
4	100	CHINA
…		

Sampler

Prob.
0.01
0.02
0.01
0.05
…

Sample size n

Item	Price	Country	Prob.
4	100	CHINA	0.05
2	50	CA	0.02
1	0	CHINA	0.01

Estimator: $q(S) = \frac{1}{n}\sum_{i=1}^{n}\frac{y_i}{p_i}$

AQP: when Multiple Samplers Available

SELECT SUM(Price) WHERE Country = ‘CHINA’

Multiple Samplers: One Size Does Not Fit All

Data

Item	Price	Country
1	10	CHINA
2	50	CA
3	300	CHINA
4	100	CHINA
…		

Uniform sampler: each tuple sampled with equal prob.

Optimized for COUNT

Prob.
0.01
0.01
0.01
0.01
…

Measure-biased sampler: prob. is proportional to measure values

Optimized for SUM

Prob.
0.001
0.005
0.03
0.01
…

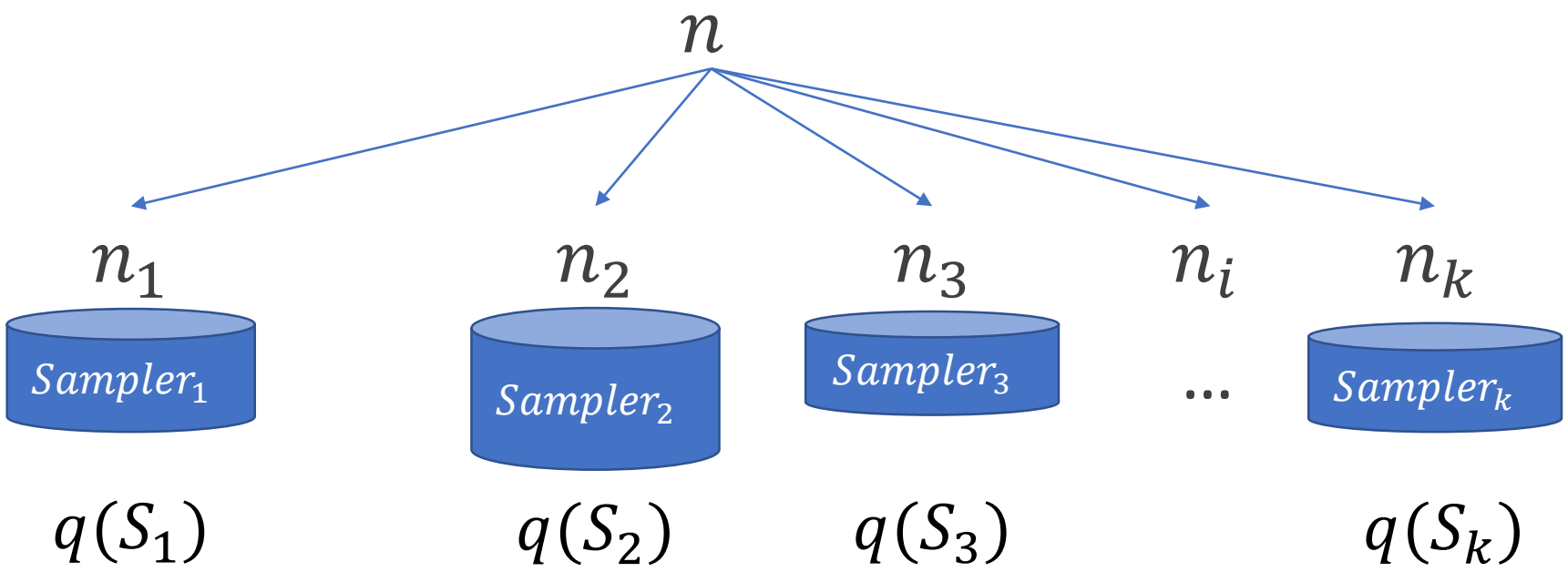
Stratified sampler: prob. is equal within each group

Optimized for GROUP-BY

Prob.
0.02
0.01
0.02
0.01
…

- Which sampler(s) should we use?
- How should we combine them?

Sampler Combination Problem (SamComb)

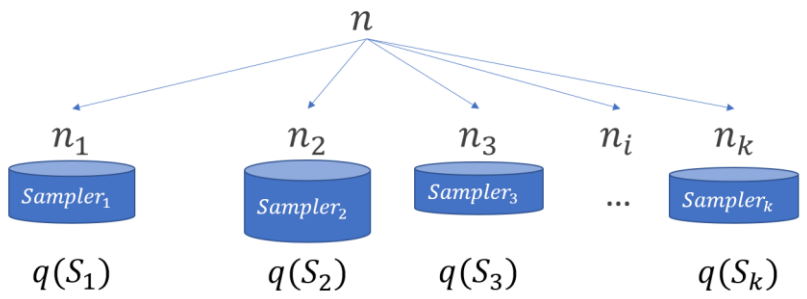


Estimator: $q(\psi) = w_1 q(S_1) + \cdots + w_k q(S_k)$

Goal: minimize $\mathrm{var}(q(\psi))$

Variables: weights $w_1, w_2, \ldots, w_k$ s.t. $\sum_i w_i = 1$;
samples $S_1, S_2, \ldots, S_k$

SamComb: two Sub-Problems



Estimator: $q(\psi) = w_1 q(S_1) + \cdots + w_k q(S_k)$

Goal: minimize $\mathrm{var}(q(\psi))$

Variables: weights $w_1, w_2, \ldots, w_k$ s.t. $\sum_i w_i = 1$;
samples $S_1, S_2, \ldots, S_k$

Budget allocation: how to allocate sample budget $n_i = |S_i|$ for each sampler?

Weight allocation: given samples, how to allocate weights w_i in the estimator?

Optimal allocation

$$w_i = \frac{n_i}{var(D_i)} / \sum \frac{n_j}{var(D_j)}$$

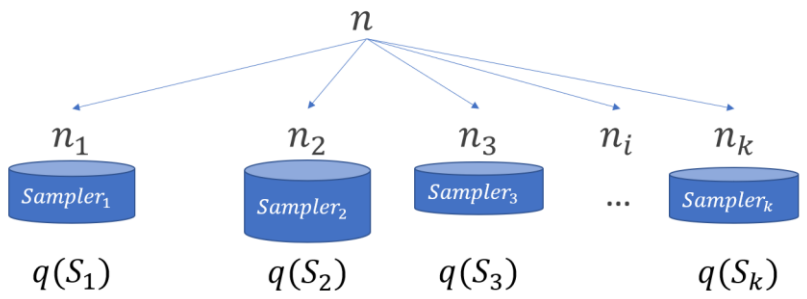
Pseudo-optimal allocation

$$w_i \sim \frac{n_i}{\widehat{var}(D_i)}$$

Prop-to-size allocation

$$w_i \sim n_i$$

SamComb: two Sub-Problems



Estimator: $q(\psi) = w_1 q(S_1) + \cdots + w_k q(S_k)$

Goal: minimize $\mathrm{var}(q(\psi))$

Variables: weights $w_1, w_2, \ldots, w_k$ s.t. $\sum_i w_i = 1$;
samples $S_1, S_2, \ldots, S_k$

Budget allocation: how to allocate sample budget $n_i = |S_i|$ for each sampler?

Weight allocation: given samples, how to allocate weights w_i in the estimator?

The optimal budget allocation is to allocate all the budget to the best sampler (i.e., with minimal $var(D_i)$).

Don't know the best sampler in advance?

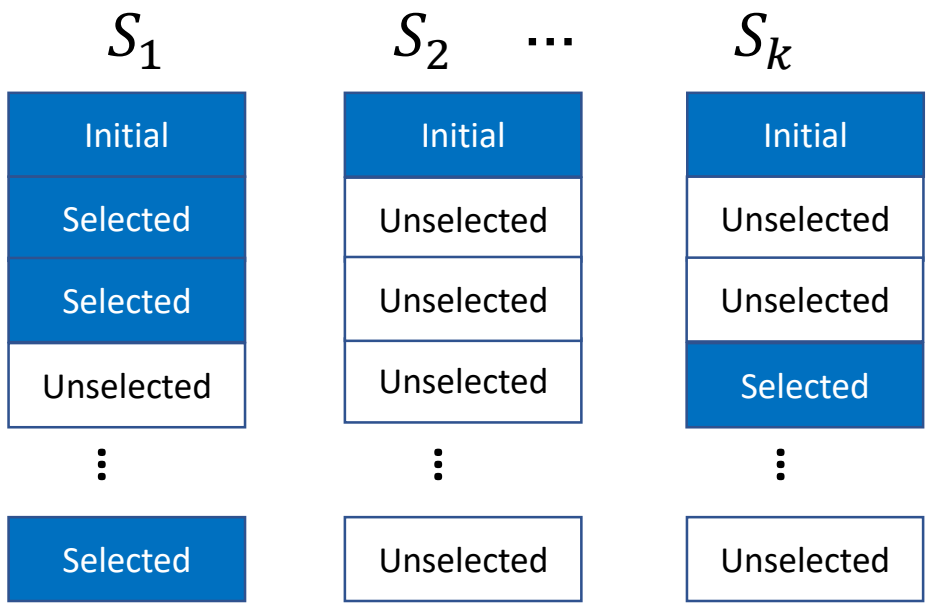
Use some budget to estimate the sampler quality

Model as multi-armed bandit (MAB) problem

SamComb-MAP: Exploration & Exploitation

Initialization Phase

Allocation Phase



Exploration

use budget to explore the quality of each sampler

Exploitation

allocate budget to the empirically best sampler

Combination

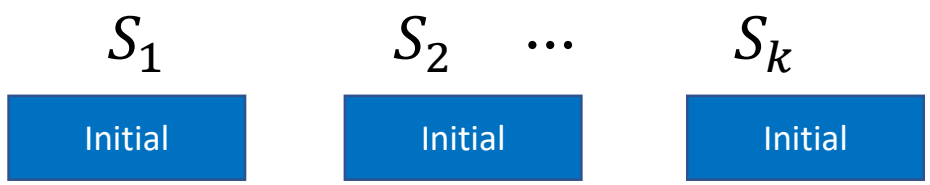
$$q(S) = w_1 q(S_1) + \ldots + w_k q(S_k)$$

SamComb-MAP: Exploration & Exploitation

Initialization Phase

Allocation Phase

ε_t-Greedy and LCB are asymptotically optimal with prop-to-size weight allocation



ε_t-Greedy:

- Select a random sampler with prob ε_t
- Select the empirically best sampler with prob $1 - \varepsilon_t$

a sub-optimal sampler is allocated $O(\ln n)$ budget

Lower Confidence Bound (LCB):

- Select the sampler with the minimum lower confidence bound of the sampler's estimation error

Adaptive-LCB

- Adaptively adjust the bound based on the sample information

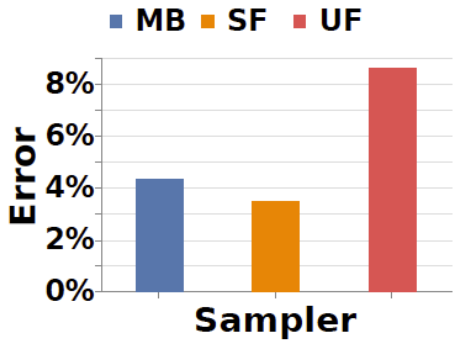
Combination

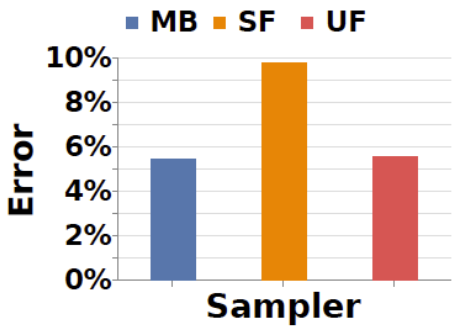
$$q(S) = w_1 q(S_1) + \ldots + w_k q(S_k)$$

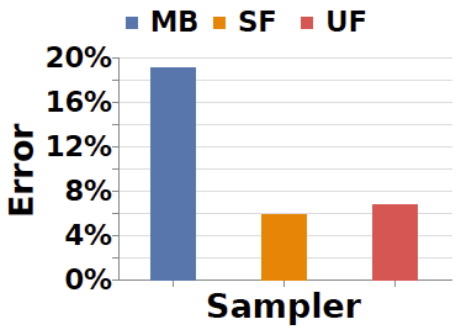
Experiments

- TPCH-Skew benchmark:

one size does not fit all



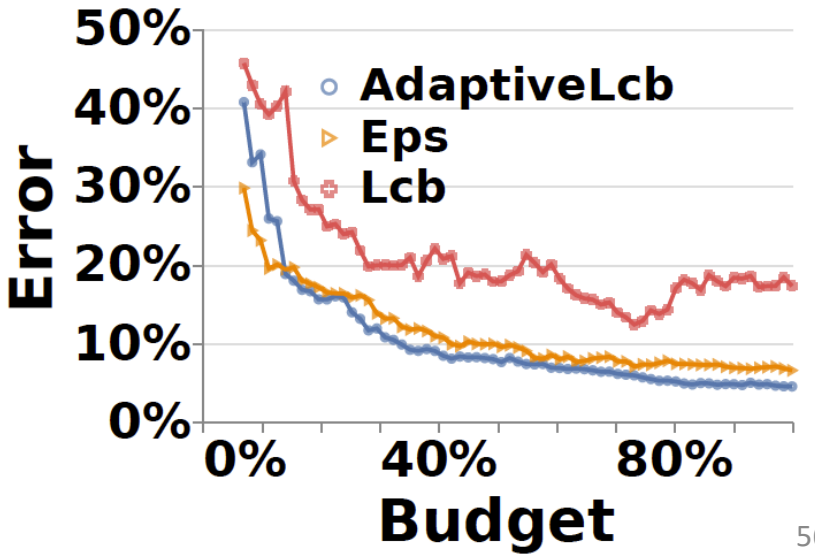




- SamComb strategy:

adaptively combine multiple samplers to improve the estimation quality

allocation + combination: no budget waste



Outline

- Learning to index [a brief survey and analysis]
- Learning to be a statistician: NDV estimation [VLDB2022]
- Learning to optimize: a case for AQP [SIGMOD2022]
- Deployment: Baihe (百合) framework

Actual Deployment: Challenges

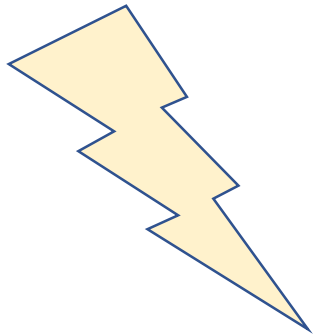
- ML4DB sounds great, but...
  - It can be challenging to deploy in a real world production setting
  - Even more so in mission critical systems such as DBs!

- Key Issues
  - Brittleness of most models, hard to detect stochastic failures, model training is not a well-defined process
  - Dependencies (ML stacks, external services, ...)
  - Keeping track of training data and model versions

DBs and ML: A Tale of Two Cities

Databases

- Central and fundamental piece of IT infrastructure for almost any business
- Need rock-solid, reliable and stable behavior
- Deterministic



Machine Learning

- Produces predictions based on probabilistic methods
- Needs close supervision due to wide range of failure modes
- Stochastic

Need to resolve conflict through proper software engineering practices!

→ Requirements, design iterations, …

Deriving Requirements

High Level Design Philosophy

- Separation from the core system
- Minimal third party dependencies

- Robustness, stability and fault tolerance
- Usability and configurability.

Concrete ML-related requirements

- Supporting wide range of models and ML frameworks
- Supporting iterative model development and evaluation process

- Not introducing too many dependencies
- Training outside of the core system
- Fallbacks in case of model failures

- Well defined and robust model deployment procedure

Different User Perspectives

Actual User (both human and technical)

- Should not have to care about what have happened inside (with ML4DB)

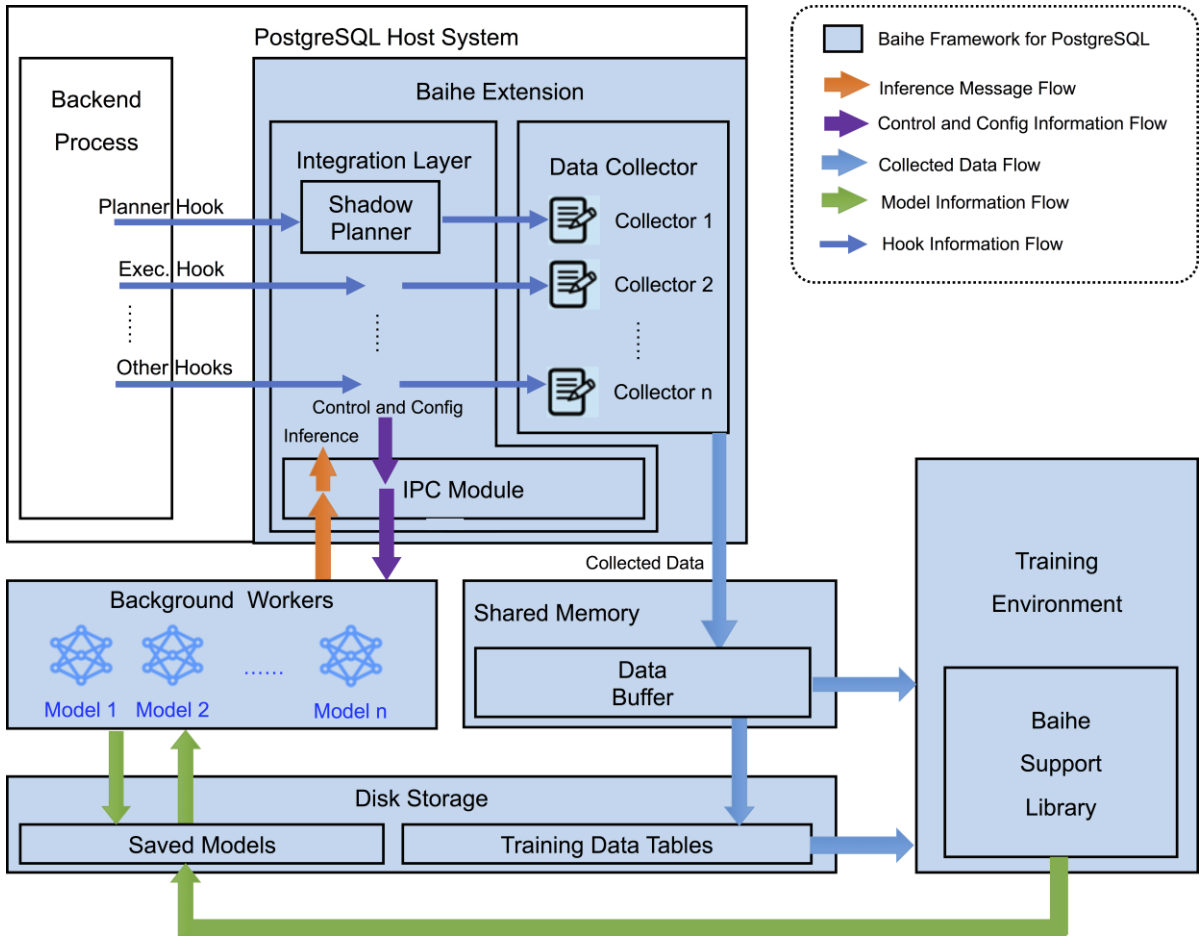
DBAs/OPs

- Should only have to learn a few config knobs and management commands (e.g. activate training data collection, configure which model to use, deploy a readily packaged model,…)
- Should not be responsible for managing extra external services which provide inference APIs
- Should not have to worry that some fancy new piece of tech breaks their system or security.

Model Developer / Researcher

- Easy access to training data, fast development iteration cycles
- Develop models based on environments (libs, packages, …) close to the usual ML/DS stack
- Simple packaging and deployment (resp. handover to DBA)

Baihe: Design Blue Print and Implementation



Highlights

- Extend Postgres **hooking mechanism** through **shadow planner** component
- Make idiomatic use of Postgres functionality for **shared memory**, **background workers**, **process management**, and **IPC**

- Simple model packaging and deployment for cardinality estimation, query runtime prediction, query optimization, and more… (to-do: reinforcement learning)

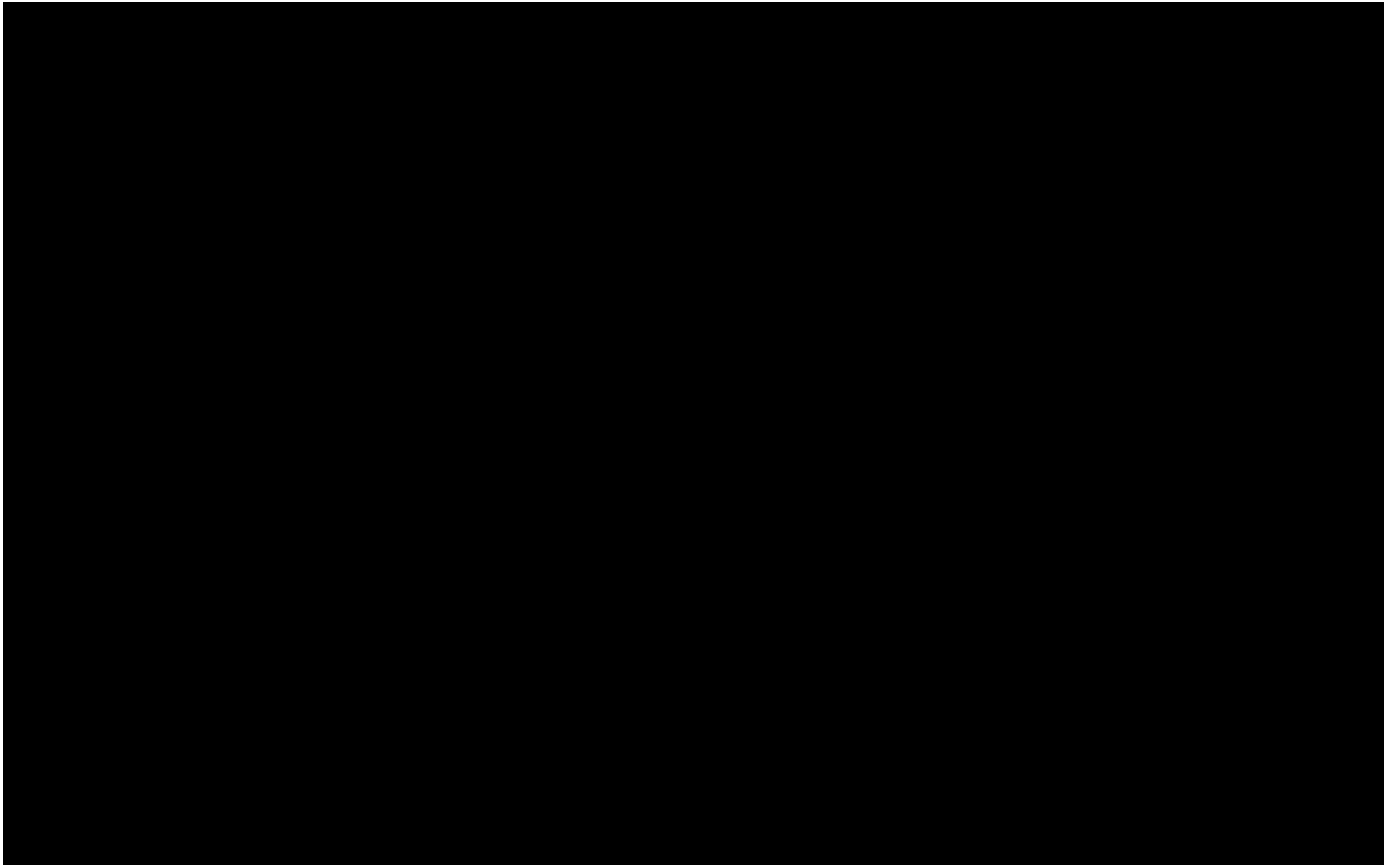
- Easy to use test bed for ML4DB research

Open Source Release

- Research oriented MVP: available soon!
- Comments and discussions welcome!

Demo Time!

- Andreas Pfadler, Rong Zhu, Wei Chen, Botong Huang, Tianjing Zeng, Bolin Ding, Jingren Zhou. *Baihe: SysML Framework for AI-driven Databases*, *Arxiv*, 2022. https://arxiv.org/abs/2112.14460



Outline

- Learning to index [a brief survey and analysis]
- Learning to be a statistician: NDV estimation [VLDB2022]
- Learning to optimize: a case for AQP [SIGMOD2022]
- Deployment: Baihe (百合) framework

智能系统团队研究方向

Lab members and Research interns

(hiring contact: bolin.ding@alibaba-inc.com)

数据隐私

- 隐私法规监管和隐私技术
- 从算法技术、系统、到应用
- 隐私机制、市场效率、和社会伦理

智能系统

- AI4Database: 智能助力数据系统
- AI4AI: 智能助力机器学习系统
- AI4Econ: 智能助力营销策略