Machine learning and databases

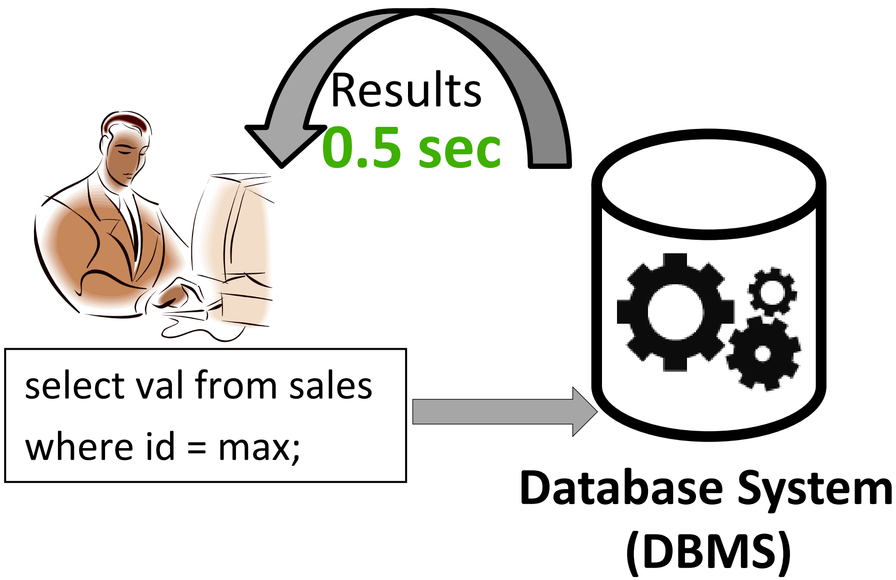
Friends or foes?

Renata Borovica-Gajic

renata.borovica@unimelb.edu.au
http://renata.borovica-gajic.com/

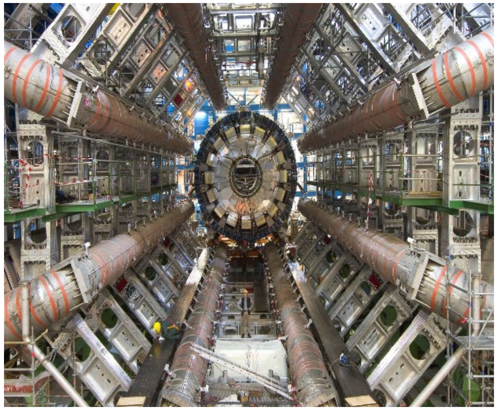
THE UNIVERSITY OF MELBOURNE

Databases = fast retrieval



Modern applications challenge status quo

Modern applications are challenging



Properties:

- Ever growing data
- Ad hoc data exploration
- Multi-tenancy

Challenges:

- Complex optimization problems
- Analytical models fail

Machine Learning (ML) to the rescue

Photos credit: Bloomberg, Stock market°, Atlas experiment, CERN*, Strato Data Centre, cloud^

Why now?

Computational power

Can adjust beyond history

Free telemetry (features)

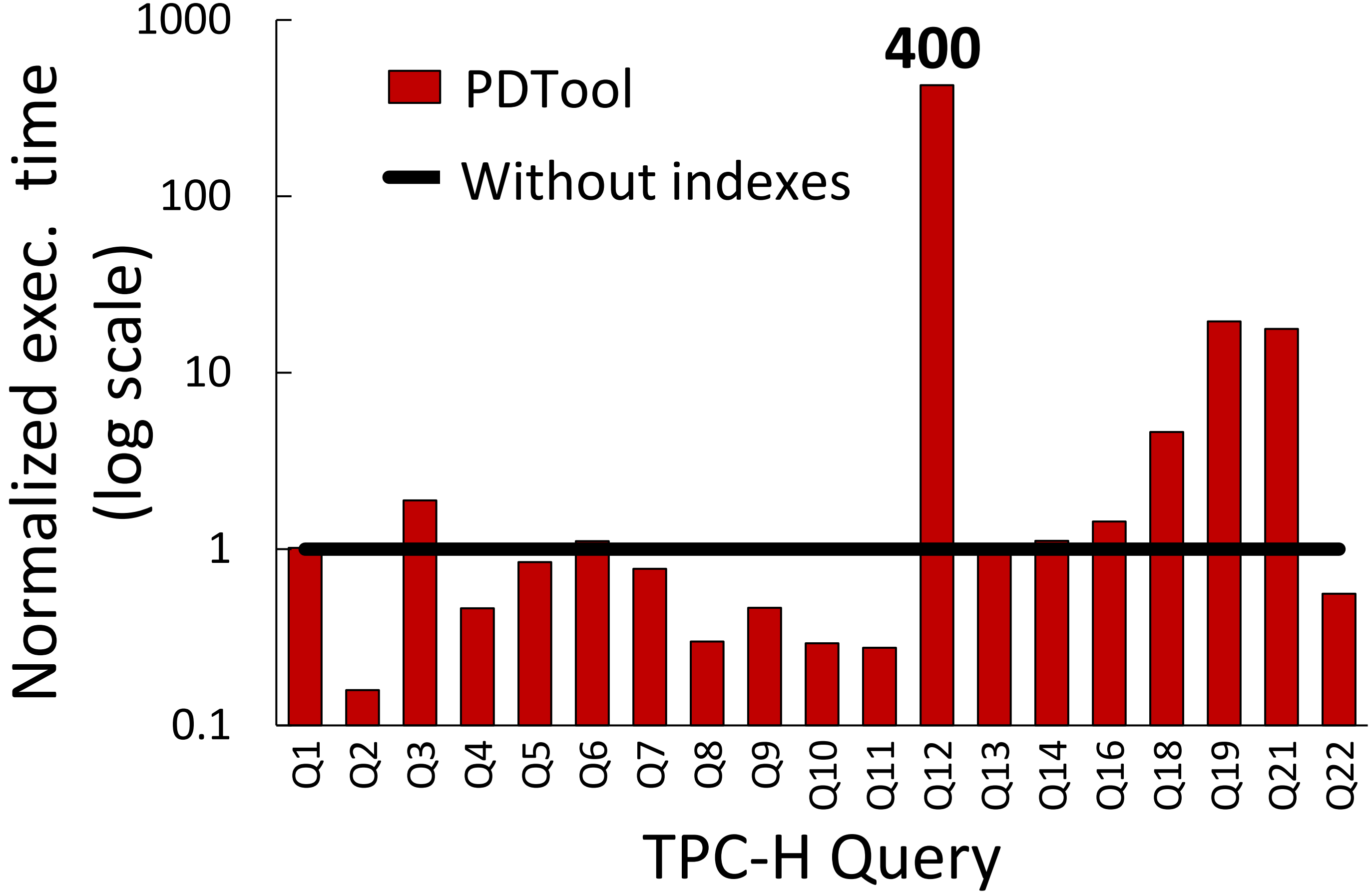
```
<RelOp AvgRowSize="35" EstimateCPU="0.0714" EstimateIO="0.081"
  <OutputList>
    <ColumnReference Database="[TPCH_001]" Schema="[dbo]" Table
      <ColumnReference Column="Expr1006" />
  </OutputList>
  <MemoryFractions Input="0.109873" Output="1" />
  <RunTimeInformation>
```

DBMS needs and ML capabilities = perfect match

Are there real use cases?

[VLDBJ'18, ICDE'15, DBTest'12]

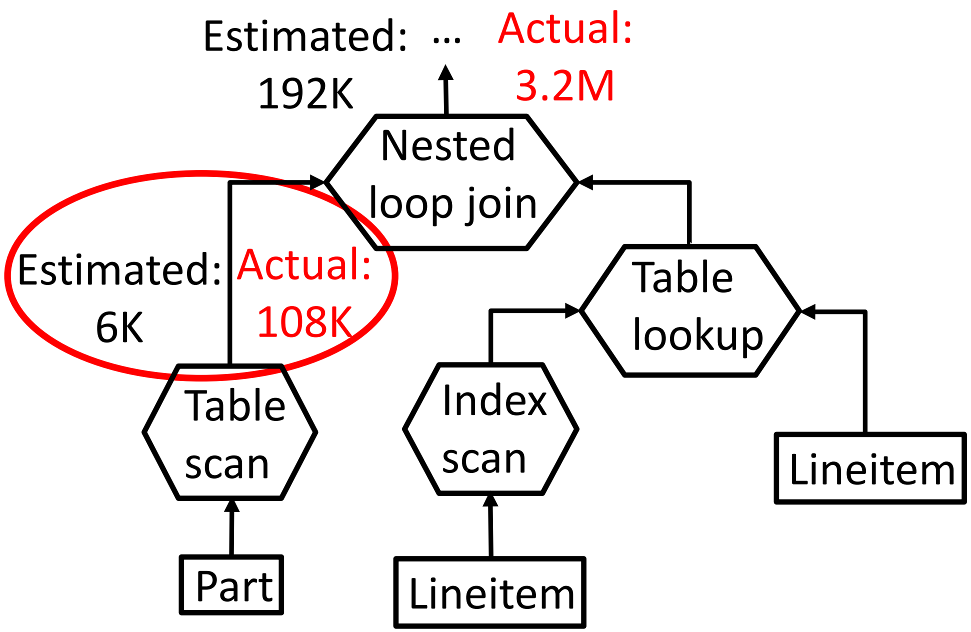
Setting: TPC-H, SF10, DBMS-X, Tuning tool (PDTool) 5GB for indexes



Plenty! Performance tuning an obvious choice

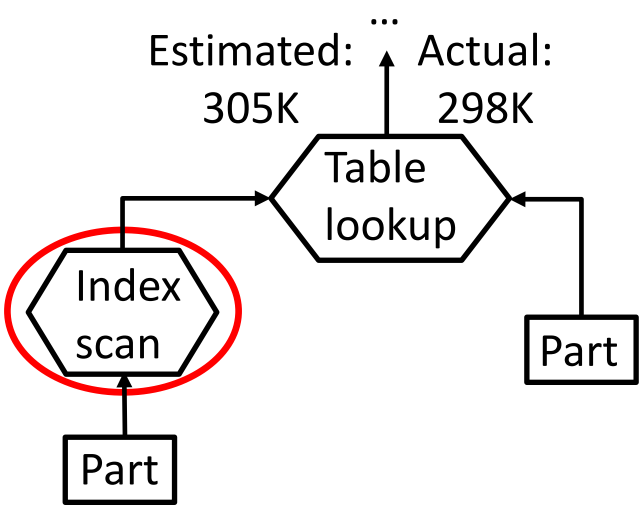
Cause for sub-optimal plans

Cardinality errors



Order of magnitude more tuples

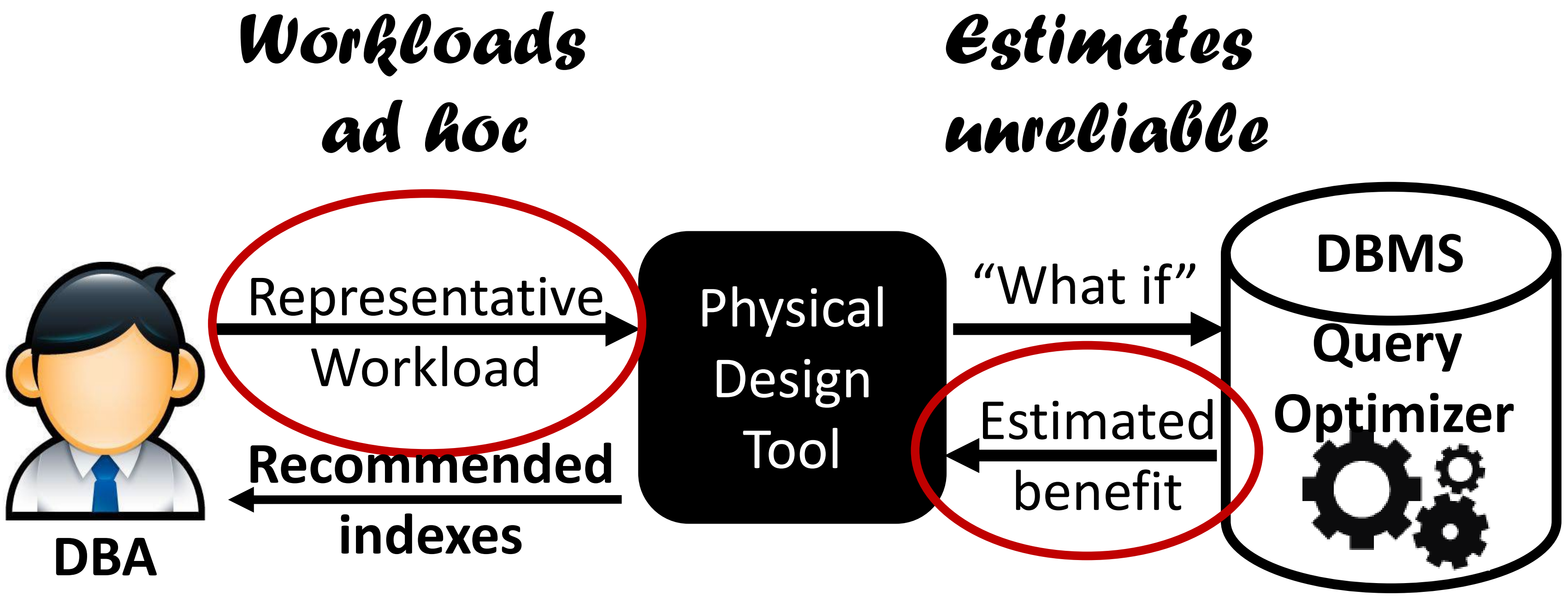
Cost model



Wrong decision of cost model

Analytical modeling is hard!

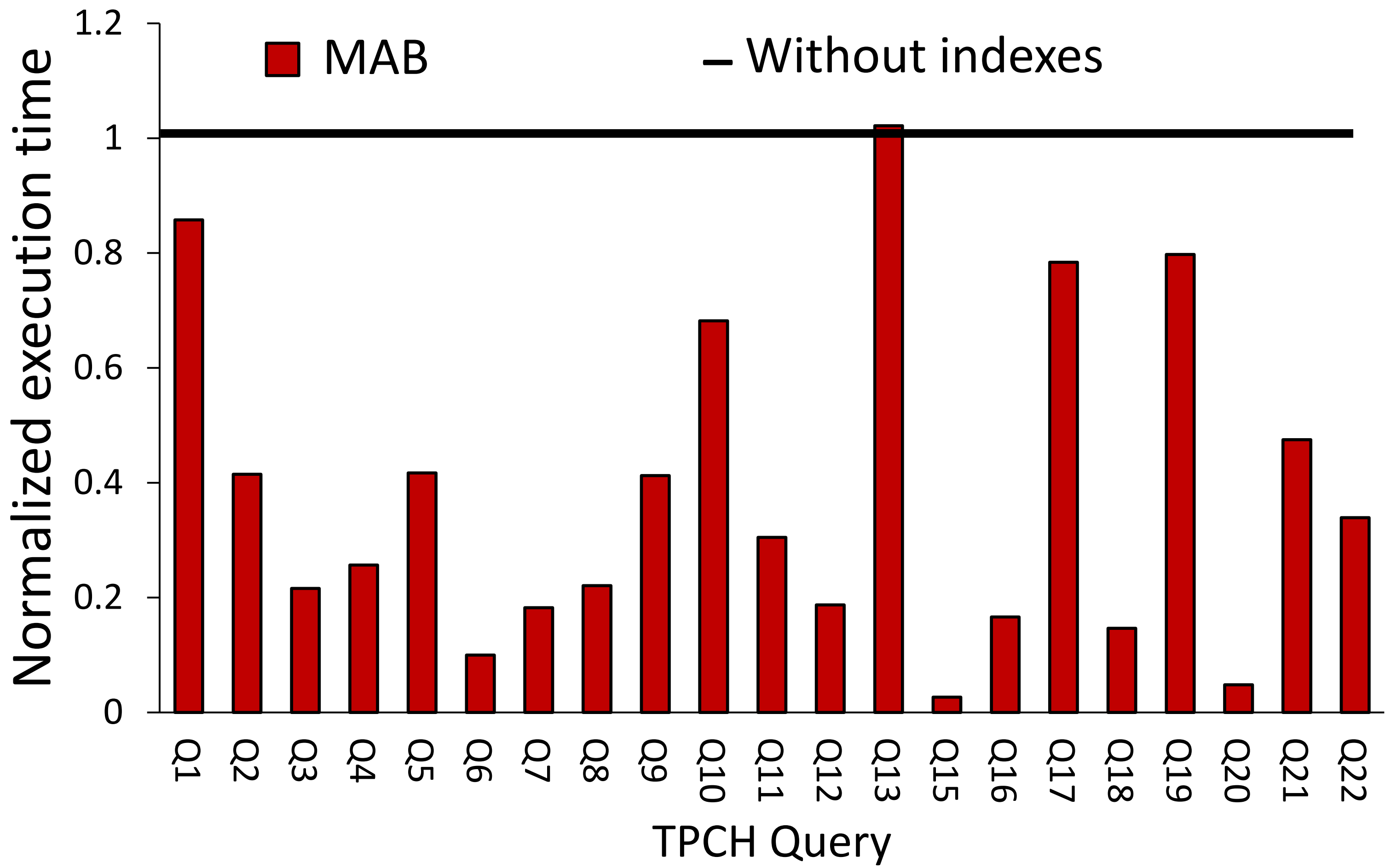
Index tuning under looking glass



Broken pipeline….

(M)Learning to the rescue

Setting: TPC-H, SF10, DBMS-X, Multi-armed bandits (MAB) for index tuning



3x Speed up vs. previous 22x slowdown

Outline

- **Performance tuning with MAB**

  [ICDE’21, ICDM’21]

- **Lightweight learned indices**

  [ADC’20]

- **Critical view on learning-based algorithms**

Outline

- **Performance tuning with MAB**

  [ICDE'21, ICDM'21]

- **Lightweight learned indices**

  [ADC'20]

- **Critical view on learning-based algorithms**

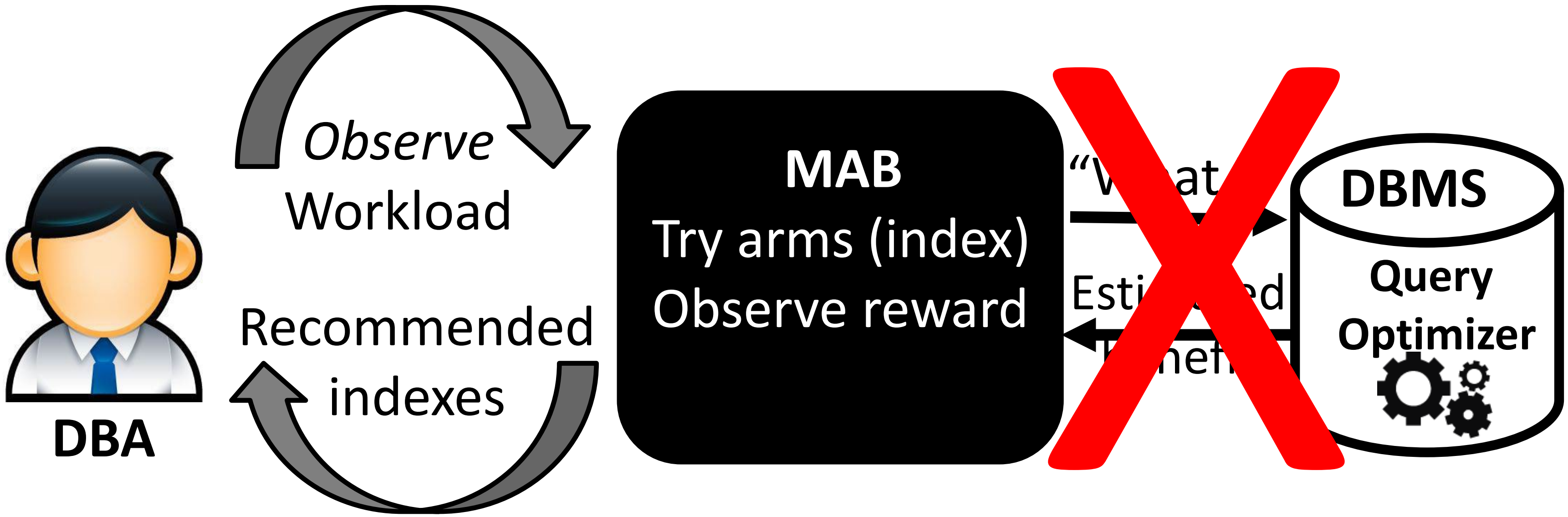
Learning with Multi-armed bandits (MAB)

- Pull an arm (slot machine) observe a reward (win/lose)
- Explore vs exploit
- Find a sequence of arms to maximize reward
- Many variants, but C^2UCB most interesting

Optimism in the face of uncertainty

Index tuning with MAB (C²UCB)

[ICDE'21]



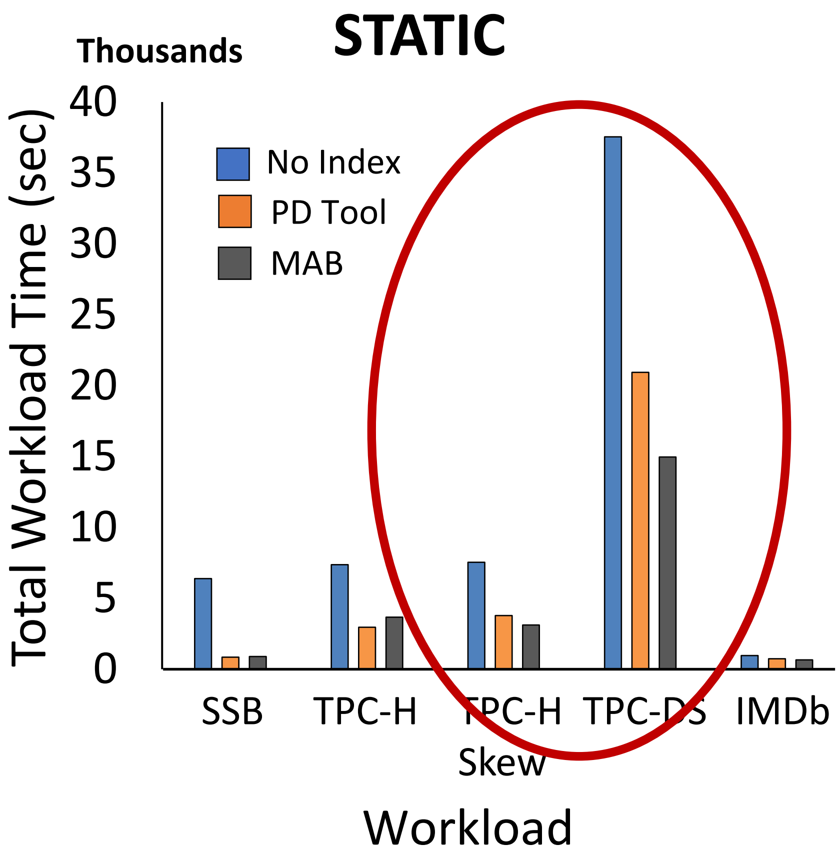
- **UCB** *guarantees* to converge to optimal policy
- **C** *(contextual)* learns benefit of arms *without* pulling them
- **C** *(combinatorial)* pulls a set of arms per round given constraints

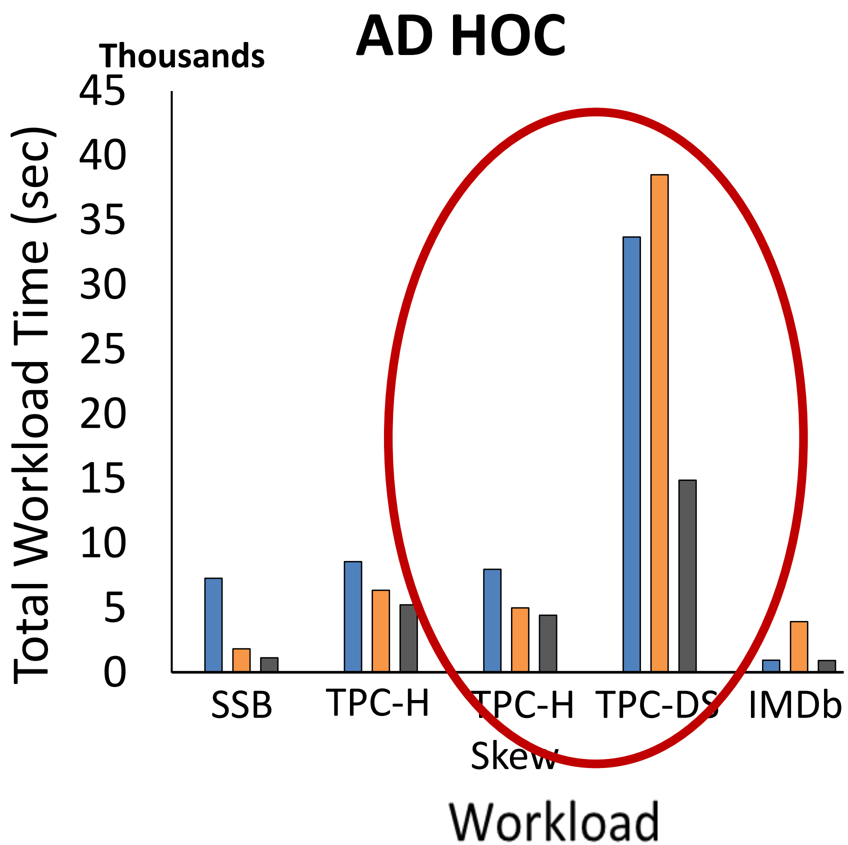
Safety guarantees with fast convergence

MAB in action

[ICDE'21]

Setting: TPCH, TPCH skew, TPC DS, SSB (10GB); IMDb(6GB) datasets static (repetitive) vs random (ad hoc) queries, MAB vs PDTool, 25 rounds





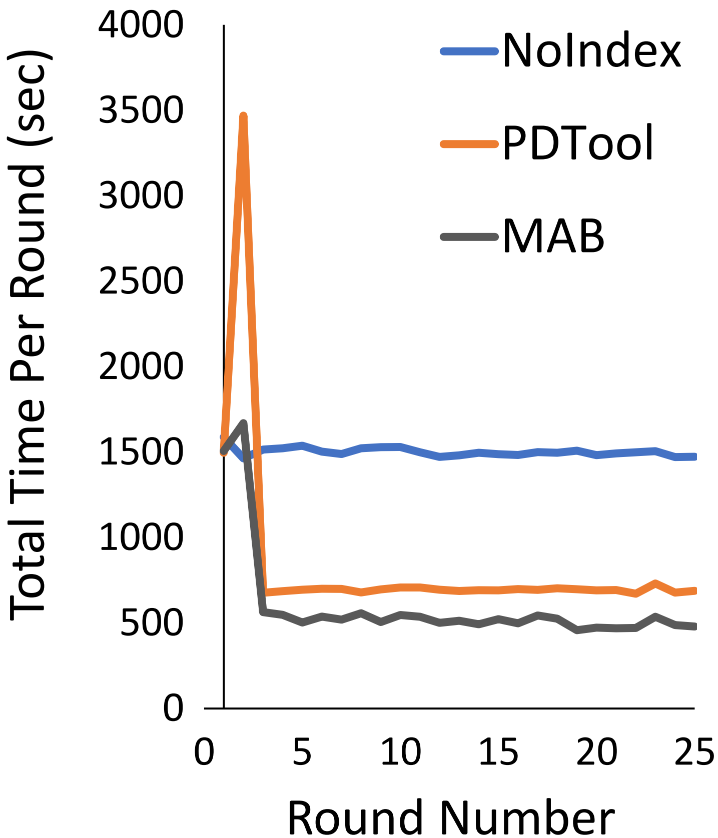
MAB robust against complex unpredictable workloads and skew

MAB in action: Zoom in TPC-DS

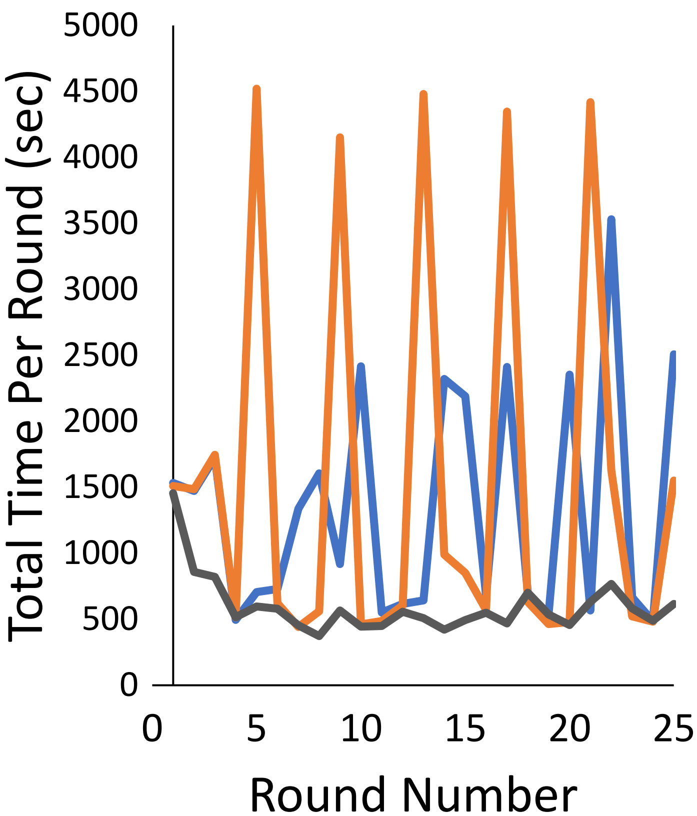
[ICDE'21]

Setting: TPC-DS, static vs ad hoc queries, MAB vs PDTool, 25 rounds

STATIC



AD HOC

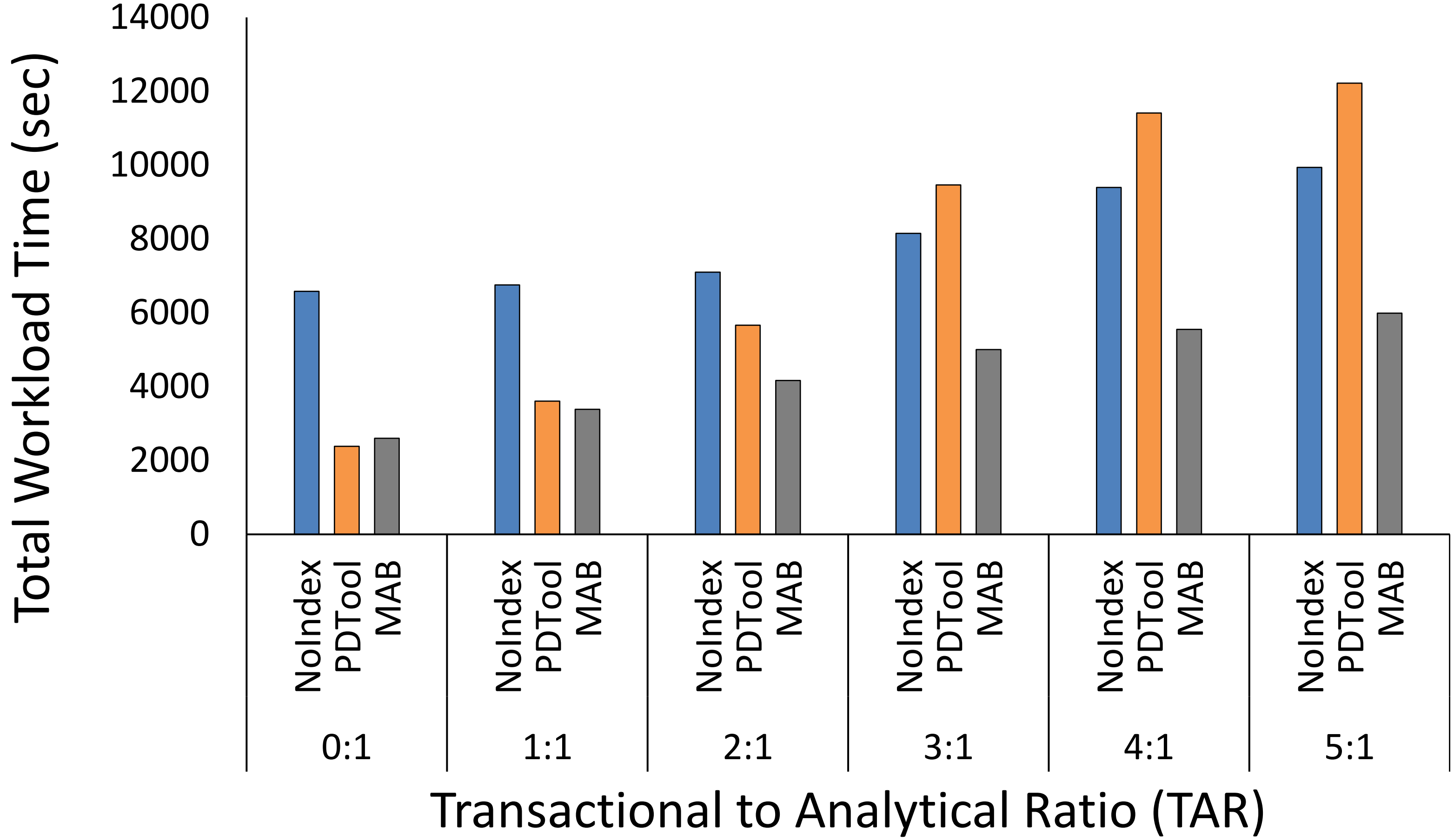


Lightweight, yet effective

Dealing with complexity (HTAP)

[under submission]

Setting: CH-BenCHmark under static workloads, MAB vs. PDTool, 25 rounds

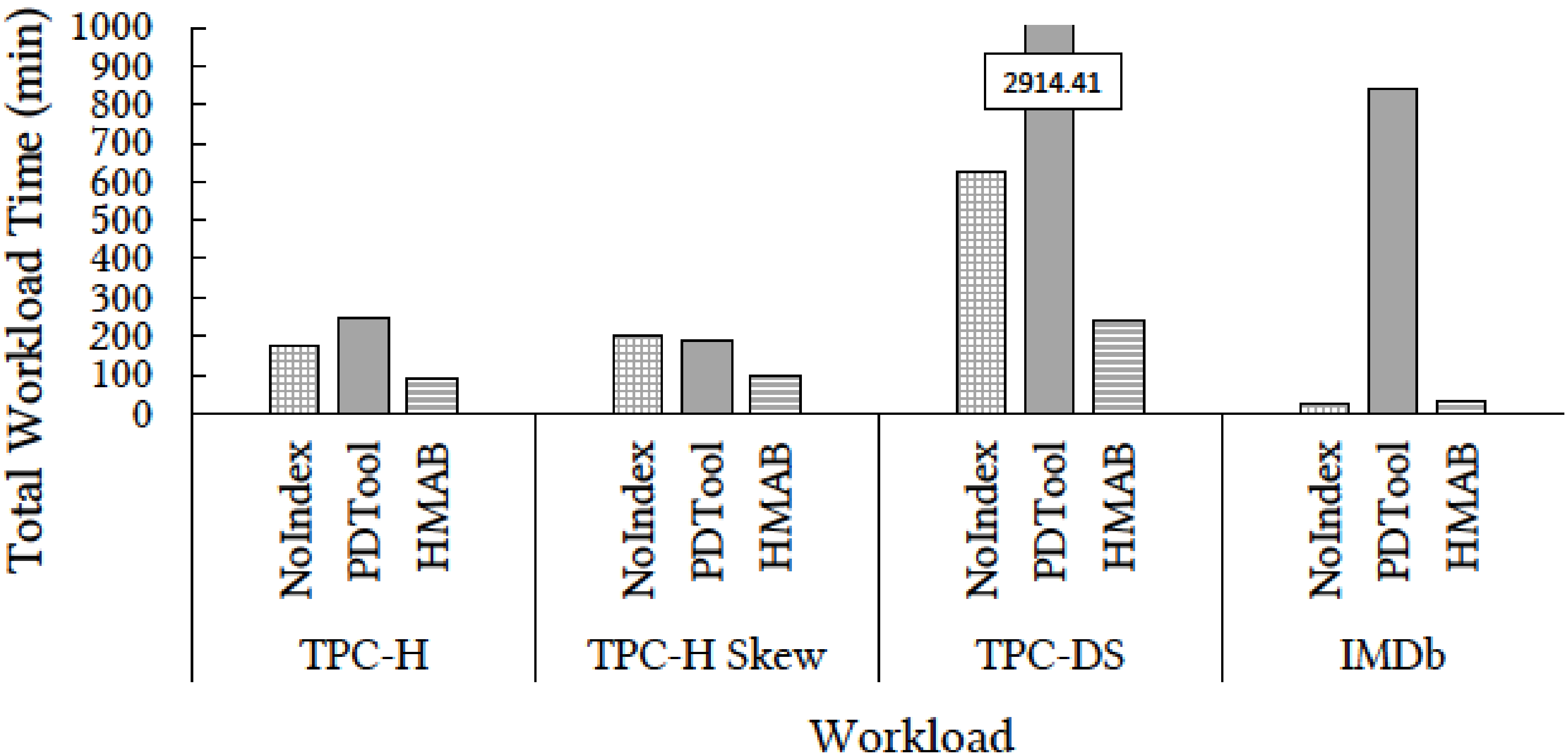


MAB adapts to complex environments

Dealing with complexity (indexes & views)

[under submission]

Setting: indexes & mat. views, dynamic workloads, MAB vs. PDTool, 25 rounds



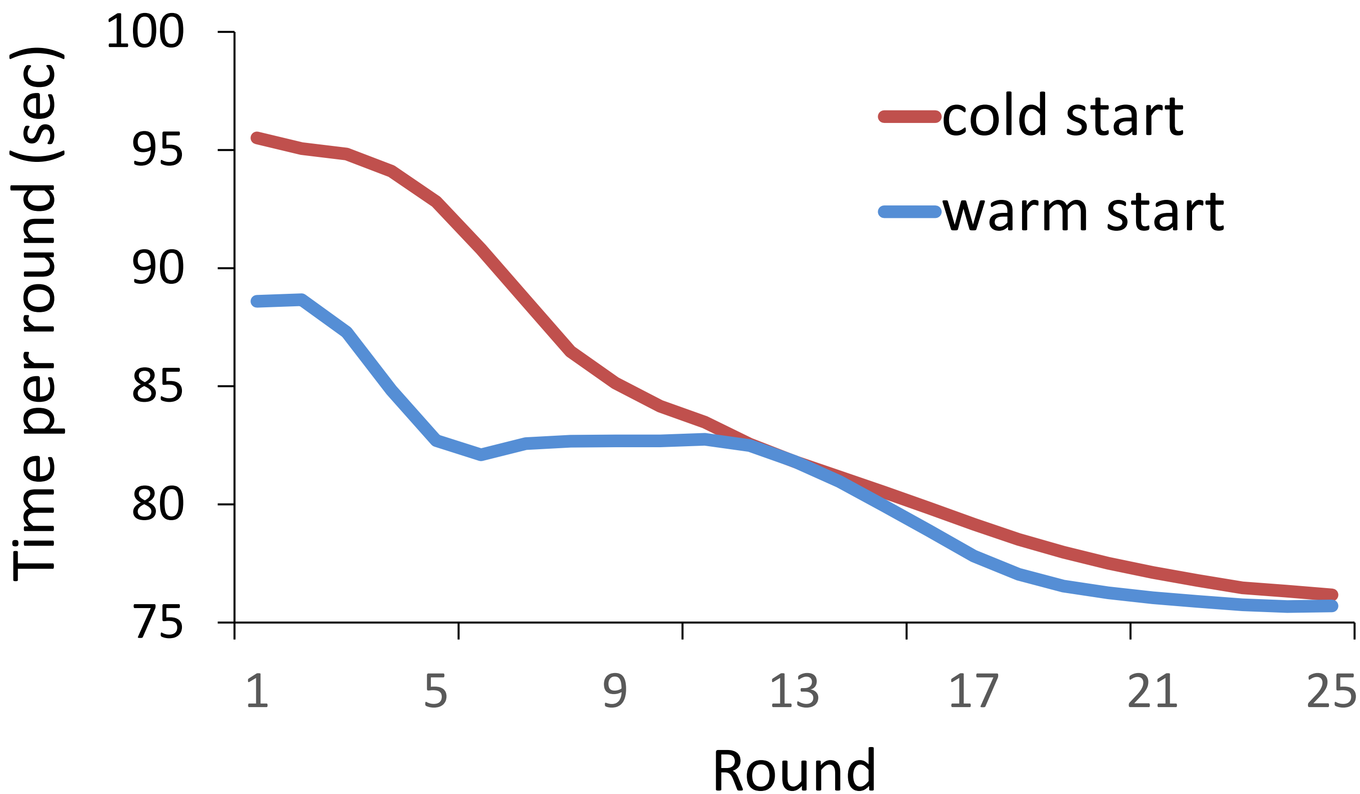
MAB adapts to heterogenous data structures

But isn't exploration too expensive?

Cutting to the chase with warm bandits

[ICDM'21]

Setting: TPC-H benchmark 10GB, 5 queries, 25 rounds *static*



(Inexpensive) warm up reduces exploration cost

Performance tuning with MAB

Summary

- MAB is a lightweight solution for physical design tuning
- C^2UCB enables exploration *without* pulling all arms
- Safety bounds guarantee convergence to optimal choice (in hindsight)
- MAB successfully deals with tuning tools' stumbling blocks (optimizer's misestimates, unpredictable workloads, HTAP, heterogenous data structures)
- Up to 96% improvement and 35% on average compared against a commercial tuning tool

Outline

- **Performance tuning with MAB**

  [ICDE'21, ICDM'21]

- **Lightweight learned indices**

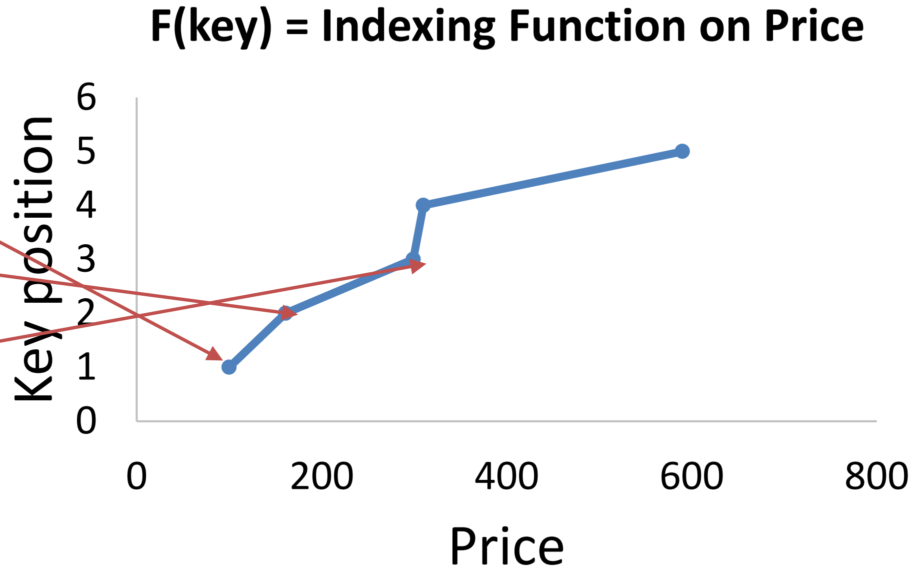
  [ADC'20]

- **Critical view at learning-based algorithms**

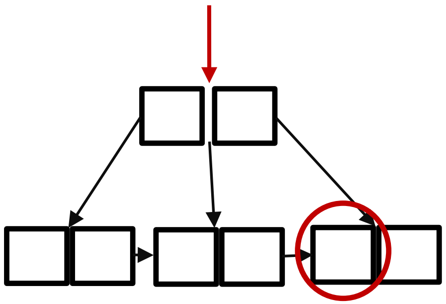
Mathematical view on indexing

I(price)

Position	Product	Price (Key)
1	Product A	100
2	Product X	161
3	Product L	299
4	Product D	310
5	Product G	590

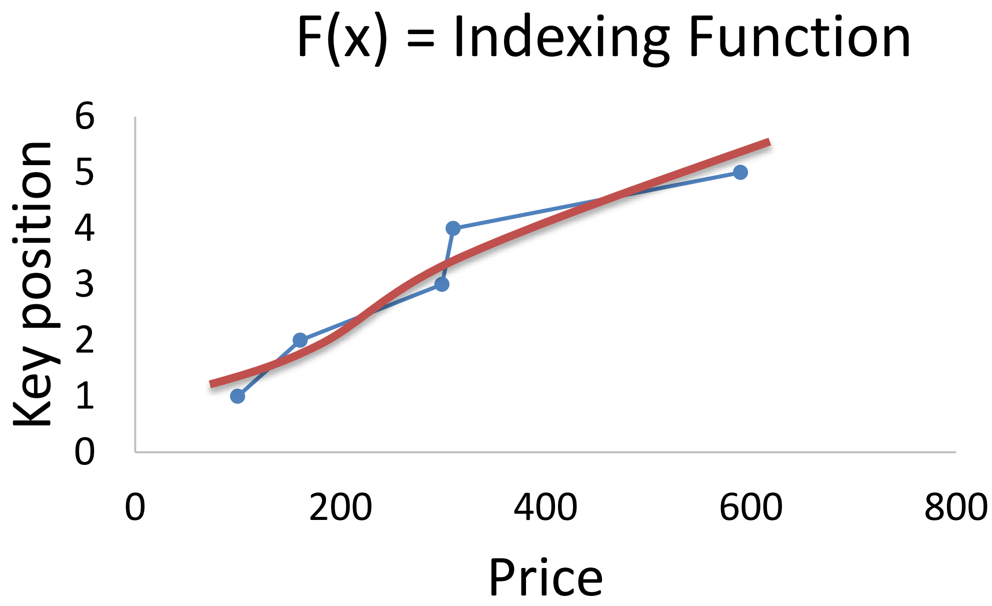


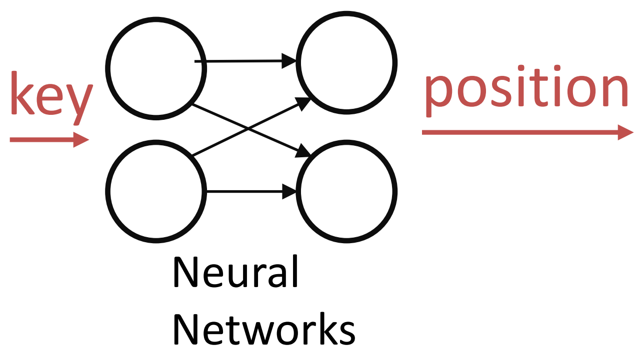
An index is a **function** $f: U \mapsto N$ that **takes** a **key** and **returns its position.**

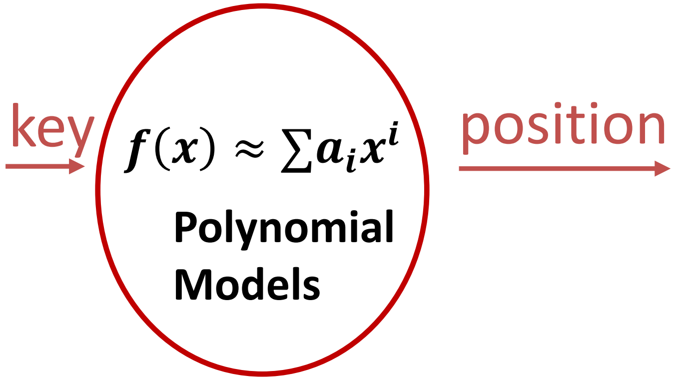


Keys form monotonically increasing CDF

So... we can build a model to predict them!







Kraska et al.[SIGMOD'18]

Learned index as a function approximation

[ADC'20]

For a chosen degree n

$$position \approx a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

Coefficients given by Discrete Chebyshev Transform

$$\alpha_i = \frac{p_i}{N} \sum_{k=0}^{N-1} \left[f\left(-\cos\left(\frac{\pi}{N}\left(k + \frac{1}{2} \right) \right) \right) \cdot \cos\left(\frac{i\pi}{N}\left(N + k + \frac{1}{2} \right) \right) \right]$$

$$p_0 = 1, p_k = 2 \text{ (if } k > 0)$$

Need to store *only* coefficients…

Function interpolation for learned indices

[ADC'20]

Model Type	Average query time (nsec)			Creation time (sec)
	Normal	LogNormal	Uniform	
B-Tree	31.5	46.0	56.3	34.6
Function interpolation (Chebyshev Polynomials)	62.1	751	40.2	3.8
Neural Network Model	402	1100	516	1 hour

Model Type	Size of Database (in Entries)			
	500k Entries	1M Entries	1.5M Entries	2M Entries
B-Tree	33.034 MB	66.126 MB	99.123 MB	132.163 MB
Neural Network	210.73 kB	210.73 kB	210.73 kB	210.73 kB
Chebyshev Polynomials	1.8kB	1.8kB	1.8kB	1.8kB

30-90% faster at querying than NN, 99% space saving

Function interpolation to the rescue

Summary

- Use of simple function interpolation instead of NN for learned index approximation
- Benefits:
  - No hyperparameter tuning
  - Fast creation time (10x)
  - Higher compression rate (99% space saving)

Outline

- **Performance tuning with MAB**

  [ICDE’21, ICDM’21]

- Lightweight learned indices

  [ADC’20]

- **Critical view on learning-based algorithms**

Properties for future DBMS adoption

- **Small computational overhead**
  - Pre-training important, yet often ignored
  - Resources plus time invested
- **Ability to adapt and generalize**
  - See the past, adjust to unpredictable future
  - Train on development port to product environment
  - Transfer learning critical
- **Safety guarantees required**
  - Prove it does the right thing
  - Explain the output (decisions made)

Lightweight, yet (provably) accurate is key

Numerous opportunities for innovation

- **ML within the DB Engine**
  - Physical database design
  - Learned vs traditional data structures
  - Configuration tuning
  - Resource management
  - Query optimization
- **Innovation in ML domain**
  - Hierarchical MABs (infinite arms)
  - Pretraining for faster convergence (warm start)
  - Lightweight transfer learning

Where to go from here

"It is not the strongest species that survive, nor the most intelligent, but the ones most responsive to change." Charles Darwin

Queries
[SIGMOD'12]
[VLDB'12]
[CACM'15]
[ICDE'21]
[ICDM'21]

Data
[DBTest'12]
[ICDE'15]
[VLDBJ'18]
[ADC'20]

Hardware
[VLDB'16]
[ADMS'17]
[CACM'19]

DBMS System

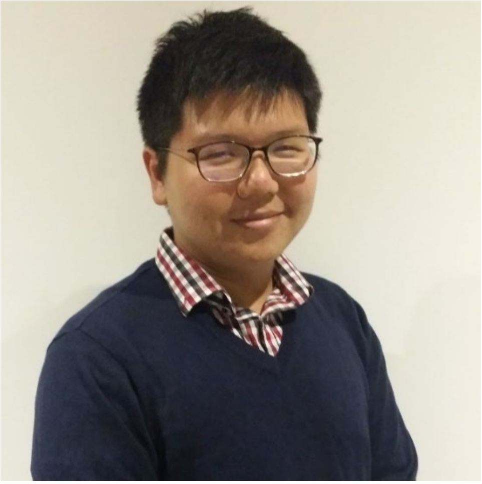
Fast response

Thank you!

Learning DBMSs for efficient data analysis

Special thanks to

Malinga Perera



Bastian Oetomo

Ben Rubinstein

Questions?

THANK YOU

BACKUP

MAB against other baselines

Setting: TPC-DS benchmark 10GB, 25 rounds *static, total time in min*

	TPC-DS			
	Rec.	Cre.	Exec.	Total
DBAB	1.47	12.86	262.88	277.21
PDTool	16.39	3.8	277.22	297.41
HMAB	1.14	7.76	219.98	228.88
Anytime	39.88	7.29	308.47	355.64
AutoAdmin	28.99	4.94	273.87	307.8
DB2Advis	0.09	4.27	279.97	284.33
Dexter	9.22	1.86	674.06	685.14
Drop	56.35	0.34	694.39	751.08
Extend	9.49	3.41	702.73	715.63
Relaxation	567.39	4.3	365.38	937.07