A tale of learning databases

Dr Renata Borovica-Gajic

renata.borovica@unimelb.edu.au
http://renata.borovica-gajic.com/
**Big data proliferation**

The Digital Universe: 50-fold Growth from the Beginning of 2010 to the End of 2020

"**Big data** is when the current technology does not enable users to obtain *timely, cost-effective, and quality* answers to *data-driven questions.*“  [Steve Todd, Berkeley]

Technology follows Moore’s Law


† “Trends in big data analytics“, 2014, Kambatla et al
What business analysts want

- Timely, predictable, cost-effective queries

80% accessed within 3 hours

- Minimal data-to-insight time
- Predictable response time
- Low infrastructure cost

[WinterCorp, 2013]
As traditional DBMS rely on predefined assumptions about workload, data and storage, changes cause loss of performance and unpredictability.

Insight

Query execution must adapt and learn form workload, data and hardware to stabilise and optimise performance and cost.
Outline

• Minimise data-to-insight time
  – *Workload-driven* learning and adaptation
    [CACM’15, SIGMOD’12, VLDB’12]

• Improve predictability of response time
  – *Data-driven* learning and adaptation
    [VLDBJ’18, ICDE’15, DBTest’12]

• Reduce analytics cost
  – *Hardware-driven* learning and adaptation
    [CACM’19, ADMS’17, VLDB’16]
Outline

• Minimise data-to-insight time
  – *Workload-driven* learning and adaptation
    
    [CACM’15, SIGMOD’12, VLDB’12]

• Improve predictability of response time
  – *Data-driven* learning and adaptation
    
    [VLDBJ’18, ICDE’15, DBTest’12]

• Reduce analytics cost
  – *Hardware-driven* learning and adaptation
    
    [CACM’19, ADMS’17, VLDB’16]
Need for efficient data exploration
Current technology ≠ efficient exploration

Data-to-insight time

Traditional query stack

- Loading
- Querying

Raw data querying stack

- Processing (Q1)
- Convert
- Tokenize
- Parse
- I/O

Time to first insight too long
Does not scale with data growth

Overheads too high
Optimise raw data querying stack

Raw data querying stack

- Processing (Q1)
- Convert
- Tokenize
- Parse
- I/O

Not everything needed for Q1

Let users show by asking queries

NoDB: Workload-driven data loading & tuning
PostgresRaw: NoDB from idea to practice

1. Positional indexing

- Pointers to end of tuples
- Pointers to attributes

Adjust to queries = progressively cheaper access
PostgresRaw in action

**Setting:** 7.5M tuples, 150 attributes, 11GB file

**Queries:** 10 arbitrary attributes per query, vary selectivity

Data-to-insight time halved with PostgresRaw

Per query performance comparable to traditional DBMS
Summary of PostgresRaw

- Query processing engine over raw data files
- Uses user queries for partial data loading and tuning
- Comparable performance to traditional DBMS

IMPACT

- Enables **timely data exploration with 0 initialisation**
- Decouples user interest from data growth
Lesson #1

Learn from workload to decrease data to insight time
Outline

• Minimise data-to-insight time
  – *Workload-driven* learning and adaptation
    [CACM’15, SIGMOD’12, VLDB’12]

• Improve predictability of response time
  – *Data-driven* learning and adaptation
    [VLDBJ’18, ICDE’15, DBTest’12]

• Reduce analytics cost
  – *Hardware-driven* learning and adaptation
    [CACM’19, ADMS’17, VLDB’16]
Index: with or without?

**Setting:** TPC-H, SF10, DBMS-X, Tuning tool 5GB space for indexes

Performance degraded after tuning
Access path selection problem

Statistics: unreliable advisor
Re-optimization: risky

Re-optimization
[MID’98, POP’04, RIO’05, BOU’14]
Quest for predictable execution

Removing variability due to (sub-optimal) choices
Smooth Scan

**Morph** between Index and Sequential Scan based on **observed result** distribution
Morphing mechanism

Modes:

1. **Index Access**: Traditional index access
2. **Entire Page Probe**: Index access probes entire page
3. **Gradual Flattening Access**: Probe adjacent region(s)
Morphing policy

- Selectivity Increase \(\rightarrow\) Mode Increase
- Selectivity Decrease \(\rightarrow\) Mode Decrease

\[
\text{SEL}_\text{region} \geq \text{SEL}_\text{global} \\
\text{SEL}_\text{region} < \text{SEL}_\text{global}
\]

Region snooping = Data-driven adaptation
Smooth Scan in action

Setting: Micro-benchmark, 25GB table, Order by, Selectivity 0-100%

Near-optimal over entire selectivity range
Summary of Smooth Scan

- Statistics-oblivious access path
- Uses region snooping to morph between alternatives
- Near-optimal performance for all selectivities

IMPACT

- Removes access path selection decision
- Improves predictability by reducing variability in query execution
Lesson #2

Learn from data
to reduce query response time
and improve predictability
Outline

• Minimise data-to-insight time
  – *Workload-driven* learning and adaptation
    [CACM’15, SIGMOD’12, VLDB’12]

• Improve predictability of response time
  – *Data-driven* learning and adaptation
    [VLDBJ’18, ICDE’15, DBTest’12]

• Reduce analytics cost
  – *Hardware-driven* learning and adaptation
    [CACM’19, ADMS’17, VLDB’16]
Storage is expensive for rarely accessed data

“Most firms estimate that they are only analyzing 12% of the data that they already have”  [Forrester 2014]

[WinterCorp, 2013]

[Hamilton, 2009]
Cost of tapes and (best case) latency of disks
But ONE disk group active at any point in time
Storage tiering in data centres

- Memory
- Disk
- Tape

Levels:
- Hot
- Warm
- Cold
Storage tiering in data centres

[VLDB’16, ADMS’17, CACM’19]

Storing 100TB of data
[Horison, 2015]

But can we run queries on top of it?

Can we shrink tiers to reduce storage cost?

2-tier architecture based on CSD HALVES storage cost

Cost (x1000$)

Storing 100TB of data
[Horison, 2015]

$159,641

Hot
Warm
Cold

Memory

Cold Storage

Performance
Query execution over CSD

**Setting**: virtualised enterprise datacenter, clients: PostgreSQL, TPCH 50, Q12, CSD: shared, layout: one client per group

Lost opportunity: CSD relegated to archival storage
Skipper to the rescue

Virtualised enterprise data center

**Multi-way joins:**
Opportunistic execution triggered upon data arrival

```
I/O Scheduler
```

**Object-group map:**

**Progress driven caching:**
Favors caching of objects to maximise query progress

**Novel ranking algorithm:**
Balances access efficiency across groups and fairness across clients

Network

VM1  VM2

DB1  DB2

MJoin

```
Hash
Hash
Hash
```

```
Scan A
Scan B
Scan C
```

A1  A2

B1

C1
Skipper in action

**Setting:** multitenant enterprise datacenter, clients: TPCH 50, Q12, CSD: shared, layout: one client per group

**Cost benefit without (significant) performance penalty**
Summary of Skipper

• Efficient query execution over CSD with:
  1. Rank-based I/O scheduling
  2. Out-of-order execution based on multi-way joins
  3. Progress based caching policy

• Approximates performance of HDD-based storage tier

IMPACT

• Cold storage can reduce TCO by shrinking storage hierarchy
• Skipper enables data analytics-over-CSD-as-a-service
Lesson #3

Learn from HW to reduce storage cost without sacrificing query performance
Summary

• Minimise data-to-insight time
  – Workload-driven learning
  – Load/tune as a byproduct of workload execution

• Improve predictability of response time
  – Data-driven learning
  – Transform access path gradually to fit data properties

• Reduce analytics cost
  – Hardware-driven learning
  – From plan pull-based to hardware push-based execution
Is there (M) Learning in learning DBMS?

• Many decisions can be automated (with sufficient training)
• A lot of infrastructure already exists (query monitoring, execution plans, stats)
• Finding the right “hammer” for every problem is key
• Regret bounds (provable guarantees) makes it appealing
Automated tuning with provable guarantees

• With multi-armed bandit algorithms

Workload $Q_1, Q_2, \ldots, Q_t$...

Index choices

\begin{align*}
\text{Candidates} & : [A.a] \\
\text{Bandit selection} & : [B.a] \\
& \quad [B.d] \\
& \quad [A.a, A.b]
\end{align*}

Reward (30 min)

DBMS engine

\begin{align*}
\text{Query plan} & : [A.a, A.b, A.c] \\
\text{Join} & : [A.a] \\
& \quad [A.b] \\
& \quad [A.a, A.b] \\
\text{Candidates} & : [B.a] \\
& \quad [A.c] \\
\text{Index choices} & : [A.b, A.c] \\
& \quad [B.a] \\
& \quad [B.d] \\
& \quad [A.a, A.b]
\end{align*}

\begin{align*}
\sigma (A.a > 10) & \\
\sigma (B.a < 4) & \\
\sigma (A.b = 5) & \\
A & \quad B
\end{align*}
Preliminary results

**Setting:** Micro-benchmark 100M tuples, 5 attributes, 3 queries per round (varying selectivity and attributes chosen)
“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]

**Data**
- [DBTest’12]
- [ICDE’15]
- [VLDBJ’18]

**Hardware**
- [VLDB’16]
- [ADMS’17]
- [CACM’19]

**DBMS System**

Learn
Adapt
Refine

Fast response

Thank you!

Learning DBMSs for efficient data analysis
Looking ahead

Data analysis for the masses

- Data classification
- Dynamic query plans
- Approximate answers
- Storage layouts
- HW-SW co-design

* http://reportlogix.com/reporting.html

† www.tableausoftware.com
My collaborators

Anastasia Ailamaki, EPFL & Raw Labs
Ioannis Alagiannis, EPFL
Miguel Branco, EPFL & Raw Labs
Raja Appuswamy, EPFL

Thomas Heinis, EPFL & Imperial
Satya Valluri, EPFL & Oracle
Farhan Tauheed, EPFL & Oracle

Stratos Idreos, Harvard University
Marcin Zukowski, Snowflake
Campbell Fraser, Google

Thank you!
Questions?

THANK YOU
Publications


