AI Techniques in Data Management

Gabriel Campero Durand
Goals For The Day

▷ Short overview on current areas of development
  ○ Awareness of existing systems and techniques
▷ Practical understanding of some cases:
  ○ *Learned index structures* as a type of *software 2.0*
Agenda

1. Basic Background
2. Overview of Approaches (The core part)
3. A Look into a Case:
   a. Learned Index Structures
4. Presentation of our Work in this Area + Invitation to Collaborate
This talk is not about

- Introducing Machine Learning or Deep Learning
- General-purpose processing systems (e.g. graph-based or dataflow-based)
- Database techniques for building ML platforms
- Management of ML in production
- The specialized area of statistical relational learning
- Performance aspects of training deep nets
This talk is about

- How AI techniques are being used in building systems
  - Closely related to the HTAP challenges discussed in the previous class
- An invitation to collaborate with us in exploring this field
Basic Background
### Thinking Humanly

“The exciting new effort to make computers think … *machines with minds*, in the full and literal sense.” (Haugeland, 1985)

“[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning …” (Bellman, 1978)

### Thinking Rationally

“The study of mental faculties through the use of computational models.” (Charniak and McDermott, 1985)

“The study of the computations that make it possible to perceive, reason, and act.” (Winston, 1992)

### Acting Humanly

“The art of creating machines that perform functions that require intelligence when performed by people.” (Kurzweil, 1990)

“The study of how to make computers do things at which, at the moment, people are better.” (Rich and Knight, 1991)

### Acting Rationally

“Computational Intelligence is the study of the design of intelligent agents.” (Poole *et al.*, 1998)

“AI … is concerned with intelligent behavior in artifacts.” (Nilsson, 1998)

---

**Figure 1.1** Some definitions of artificial intelligence, organized into four categories.
ML is a set of techniques based on statistics, which automate the process of finding patterns in data, in order to make inference or deduction. It can be further categorized into supervised, unsupervised or reinforcement learning.

AI is a larger set of techniques about replicating human behavior.
AI: Supervised Learning.

- A large space of possibilities
- We live in a golden age for ML!

From:
AI: Deep learning (no hype, please)
AI: Deep learning
(no hype, please)
AI: Deep learning (no hype, please)
Pros

○ More flexible (though perhaps less optimal) than hand-crafted solutions.

○ Theoretical scalability to data and computation: Geoffrey Hinton wrote: “Deep learning is an algorithm which has no theoretical limitations of what it can learn; the more data you give it and the more computational time you provide, the better it is.”

○ Training and inference can be made embarrassingly parallel and accelerated with GPUs. Limited mem.

○ Methodological advances and technology support
AI: Deep learning (no hype, please)

▷ Cons
  ○ Hard to train
  ○ Hard to decide on optimal hyperparameter configuration
  ○ Hard to explain and debug

▷ Deep Learning is not a silver bullet, it works well for image recognition and highly structured data. Other models can outperform DL in other tasks.
The Contemporary State of A.I.
AI and the winters...

▷ Dissapointments after too grand expectations for machine translation (1960s)
▷ Abandonment of connectionism (1960s)
▷ Fifth generation computers and Expert Systems failures (1990s)
▷ ...
  ○ See: https://a16z.com/2016/06/10/ai-deep-learning-machines/
2.

An Overview of AI Techniques in Data Management
The Landscape of AI Techniques in Data Management

- Natural Language Interfaces for Databases
- ML Techniques for Data Integration and Cleaning
- ML Services to Enhance DB interaction
- Self-Managing Operational Aspects
- Self-Managing Database Internals
- In-Database ML
- ML Techniques for Implementing Database Internals
# The Landscape of AI Techniques in Data Management

<table>
<thead>
<tr>
<th>Natural Language Interfaces for Databases</th>
<th>ML Techniques for Data Integration and Cleaning</th>
<th>ML Services to Enhance DB Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. ML Techniques for Implementing Database Internals</td>
<td></td>
</tr>
</tbody>
</table>

Not covered today
Contact me if interested
Some Technological Context...

1. In-Database ML

- History of ML Analytics Systems
  - ML Sys. Abstractions
  - Scalable LA Sys.
  - RIOT-DB
  - Parameter Server
  - Last few years

- 1970s
  - SAS
  - Mid 1990s
  - “In-Database” ML
- Late 1990s to Mid 2000s
  -  
  - Parameter Server
  - Mid 2000s to Early 2010s
  - Cloud ML Services
  - ML on Dataflow Sys.
  - ML “Lifecycles”
In-Database ML

▷ Enterprises have large amounts of labeled data, stored in databases
▷ Analyzing it with ML brings business potential
▷ In-Database ML
  ○ Brings the computation to the data
  ○ Leverages optimizations from the DBMS
  ○ Could work with larger-than-memory data
In-Database ML

- One example: Apache MADLib-
  Magnetic (Accepts all data), Agile (HTAP), Deep
- Other examples: SAP PAL from Leonardo,

<table>
<thead>
<tr>
<th>User Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Driver&quot; Functions</td>
</tr>
<tr>
<td>(outer loops of iterative algorithms, optimizer invocations)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>High-level Abstraction Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(iteration controller, ...)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RDBMS Built-in Functions</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Functions for Inner Loops</th>
</tr>
</thead>
<tbody>
<tr>
<td>(for streaming algorithms)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Low-level Abstraction Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(matrix operations, C++ to RDBMS type bridge, ...)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RDBMS Query Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Greenplum, PostgreSQL, ...)</td>
</tr>
</tbody>
</table>

SQL, generated from specification
Python with templated SQL

Python

C++

Pivotal
Greenplum
In-Database ML: Under the hood

▷ Naive Matrix Multiplication in SQL

- SELECT A.i, B.j, SUM(A.val*B.val)
  FROM A, B
  WHERE A.j = B.i
  GROUP BY A.i, B.j;

i: row number, j: column number

0 vals are not stored, so it is a sparse representation

From:
Kumar et al., 2017- Data Management in Machine Learning: Challenges, Techniques, and Systems. SIGMOD.
Cohen et al., 2009- MAD skills: new analysis practices for big data. VLDB.
Naive Matrix Multiplication in SQL

An example:

\[
\begin{align*}
A &= \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \\
B &= \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} \\
A \times B &= \begin{pmatrix} 4 & 5 \\ 6 & 8 \end{pmatrix}
\end{align*}
\]

### SQL Example

```sql
SELECT A.i, B.j, SUM(A.val*B.val)
FROM A, B
WHERE A.j = B.i
GROUP BY A.i, B.j;
```

- `i`: row number, `j`: column number

0 vals are not stored, so it is a sparse representation.

---

From:
Kumar et al., 2017 - Data Management in Machine Learning: Challenges, Techniques, and Systems. SIGMOD.
Cohen et al., 2009 - MAD skills: new analysis practices for big data. VLDB.
In-Database ML: Under the hood

Naive Matrix Multiplication in SQL

An example:

\[ A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}, \quad A \times B = \begin{pmatrix} 4 & 5 \\ 6 & 8 \end{pmatrix} \]

- **SELECT A.i, B.j, SUM(A.val*B.val)**
- **FROM A, B**
- **WHERE A.j = B.i**
- **GROUP BY A.i, B.j**

i: row number, j: column number

0 vals are not stored, so it is a sparse representation

From:
Kumar et al., 2017- Data Management in Machine Learning: Challenges, Techniques, and Systems. SIGMOD.
Cohen et al., 2009- MAD skills: new analysis practices for big data. VLDB.
In-Database ML: Under the hood

▷ Improved Matrix Multiplication:

Better data format + UDFs.

\[
A(i, \text{row}), B(j, \text{col})
\]

\[
\text{SELECT } A.i, B.j, \text{dotproduct}(A.\text{row}, B.\text{col}) \text{ FROM } A, B;
\]

From:
Kumar et al., 2017- Data Management in Machine Learning: Challenges, Techniques, and Systems. SIGMOD.
Cohen et al., 2009- MAD skills: new analysis practices for big data. VLDB.
In-Database ML: Under the hood

▷ Authors show that some building blocks of ML can be computed using such UDF-based approaches:
  ○ Ordinary Least Squares
  ○ Gradient Descent
  ○ ...
CREATE TABLE patients(id INTEGER NOT NULL, second_attack INTEGER, treatment INTEGER, trait_anxiety INTEGER);

INSERT INTO patients VALUES
(1, 1, 1, 70),
(3, 1, 1, 50),
(5, 1, 0, 40),
(7, 1, 0, 75),
(9, 1, 0, 70),
(11, 0, 1, 65),
(13, 0, 1, 45),
(15, 0, 1, 40),
(17, 0, 0, 50),
(19, 0, 0, 50),
(2, 1, 1, 80),
(4, 1, 0, 60),
(6, 1, 0, 65),
(8, 1, 0, 80),
(10, 1, 0, 60),
(12, 0, 1, 50),
(14, 0, 1, 35),
(16, 0, 1, 50),
(18, 0, 0, 45),
(20, 0, 0, 60);

SELECT madlib.logregr_train(
    'patients',
    'patients_logregr',
    'second_attack',
    'ARRAY[1, treatment, trait_anxiety]',
    NULL,
    20,
    'irls'
);
In-Database ML: Interface

-- Set extended display on for easier reading of output (\x is for psql only)
\x on
SELECT * from p

-- ************* --
-- Result --
-- ************* --
coef          | [-0.36346994178187, -1.82416665239327, 0.119044916666666]
log likelihood| -9.41010298309
std err       | [3.2139766375094, 1.17107844860319, 0.0549799458269309]
z stats       | [-1.9799524145795, -0.874498248695549, 2.16527796960918]
p values      | [0.0477051870698128, 0.38184697353045, 0.0303664045046168]
ods ratios    | [0.001723376392323, 0.3591173546054954, 1.12642651228095]
condition no  | 326.001922792
num rows_processed | 20
num missing_rows_skipped | 0
num iterations     | 5
variance_covariance | [[10.3291381930637, -0.47430466519573, -0.171995901260052], [-0.47430466519573, 1.37142473272853], [-0.171995901260052, 0.47430466519573]]

-- Alternatively, unnest the arrays in the results for easier reading of output (\x is for psql only)
\x off
SELECT unnest(array['intercept', 'treatment', 'trait_anxiety']) as attribute,
   unnest(coef) as coefficient,
   unnest(std_err) as standard_error,
   unnest(z_stats) as z_stat,
   unnest(p_values) as pvalue,
   unnest(odds_ratios) as odds_ratio
FROM patients_logreg;

-- ************* --
-- Result --
-- ************* --

<table>
<thead>
<tr>
<th>attribute</th>
<th>coefficient</th>
<th>standard_error</th>
<th>z_stat</th>
<th>pvalue</th>
<th>odds_ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>intercept</td>
<td>-0.36347</td>
<td>3.2139</td>
<td>-1.9799</td>
<td>0.8477852</td>
<td>0.68172338</td>
</tr>
<tr>
<td>treatment</td>
<td>-1.02411</td>
<td>1.17188</td>
<td>-0.874498</td>
<td>0.381847</td>
<td>0.359117</td>
</tr>
<tr>
<td>trait_anxiety</td>
<td>0.119045</td>
<td>0.0546797</td>
<td>2.16528</td>
<td>0.8303664</td>
<td>1.12642</td>
</tr>
</tbody>
</table>
In-Database ML: Interface

-- Display prediction value along with the original value
SELECT p.id, madlib.logregr_predict(coef, ARRAY[1, treatment, trait_anxiety]),
    p.second_attack
FROM patients p, patients_logregr m
ORDER BY p.id;

-- *********** --
-- Result --
-- *********** --

<table>
<thead>
<tr>
<th>id</th>
<th>logregr_predict</th>
<th>second_attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>True</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>True</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>True</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>False</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>True</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>True</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>True</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>True</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>True</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>True</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>True</td>
<td>0</td>
</tr>
</tbody>
</table>
In-Database ML

▷ Some alternatives:
  ○ Backend choice
    ■ On top of DBMSs: MADLib, RIOT-DB
    ■ Inside of DBMSs: SimSQL
    ■ Alternative to DBMSs, but with some similarities: SystemML, RIOT, SciDB
  ○ Interface choice
    ■ SQL or extension: MADLib
    ■ ML-Oriented Language on top of SQL: Oracle R Enterprise, RIOT-DB

From:
Kumar et al., 2017- Data Management in Machine Learning: Challenges, Techniques, and Systems. SIGMOD.
In-Database ML

▷ Lots of open questions:
  ○ How to support *natively* embeddings?

▶ Optimization across linear algebra and relational, Feature Engineering Systems, Model selection, Factorized Processing

From:
https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/
https://wwwdb.inf.tu-dresden.de/research-projects/freddy/
2. ML Techniques for Implementing Database Internals

Based on the vision of software 2.0 by Andrej Karpathy

Software 2.0

I sometimes see people refer to neural networks as just “another tool in your machine learning toolbox”. They have some pros and cons, they work here or there, and sometimes you can use them to win Kaggle competitions. Unfortunately, this interpretation completely misses the forest for the trees. Neural networks are not just another classifier, they represent the beginning of a fundamental shift in how we write software. They are Software 2.0.

The “classical stack” of Software 1.0 is what we’re all familiar with — it is written in languages such as Python, C++, etc. It consists of explicit instructions to the computer written by a programmer. By writing each line of code, the programmer identifies a specific point in program space with some desirable behavior.

From:
https://medium.com/@karpathy/software-2-0-a64152b37c35
Software 1.0 is code we write. **Software 2.0 is code written by the optimization based on an evaluation criterion** (such as “classify this training data correctly”).

From: https://medium.com/@karpathy/software-2-0-a64152b37c35
It turns out that a large portion of real-world problems have the property that it is significantly easier to collect the data (or more generally, identify a desirable behavior) than to explicitly write the (most optimal) program.

From:
https://medium.com/@karpathy/software-2-0-a64152b37c35
ML Techniques for Implementing DB Internals

▷ In these cases:
  ○ **Software 2.0 developers**
    - curate, maintain, clean and label datasets; define the learning task, such that the program will learn via the optimization.
  ○ **Software 1.0 developers**
    - maintain the surrounding tools, analytics, visualizations, labeling interfaces, infrastructure
ML Techniques for Implementing DB Internals

▷ **Software 2.0**

○ Karpathy proposes that there are examples found in visual/speech recognition, database indexing, etc.

From: https://medium.com/@karpathy/software-2-0-a64152b37c35
ML Techniques for Implementing DB Internals

▷ **Software 2.0**
  - **Pros**
    - Final program is computationally homogeneous and amiable for a specialized hardware design
    - Constant running time + memory use
  - **Cons**
    - Limited explainability, Novel requirements for managing failures and for support to developers

▷ **Examples:**
  - Cuttlefish, (The case for learned index structures is explained later, but maybe skip to that for a more straight forward example)
ML Techniques for Implementing
DB Internals

Cuttlefish: A Lightweight Primitive for Adaptive Query Processing

Tomer Kaftan
University of Washington

Alvin Cheung
University of Washington

Magdalena Balazinska
University of Washington

Johannes Gehrke
Microsoft

Cuttlefish (2018):

- Each operator in a query plan requires a cost model
- It is difficult to expect each user to provide cost models
- *Cuttlefish* proposes to substitute operators by a learning model that explores variants and their cost.
  - Partly done before in *Microadaptivity for Vectorwise*.
- *Cuttlefish* adapts to data features, not averages

From:
ML Techniques for Implementing DB Internals

From:

```python
class Tuner(choices):
    def choose(context=None) -> (Choice, Token)
    def observe(token, reward) -> None
```
ML Techniques for Implementing DB Internals

(a) Convolution variants and a function to extract dimensions

```python
# Define the convolution variants
def loop_convolve(image, kernel) -> Image
def mm_convolve(image, kernel) -> Image
def fft_convolve(image, kernel) -> Image

# Define a func. to extract image and filter dims
def extract_dimensions(image, kernel) -> Vector

conv_choices = [loop_convolve, mm_convolve, fft_convolve]
tuner = Tuner(choices = conv_choices)
results = []

# images: the list of images to convolve
# kernel: the convolution kernel to use
for image in images:
    dimgs = extract_dimensions(image, kernel)
    convolve, token = tuner.choose(context = dims)
    start = datetime.now()
    convolved_image = convolve(image, kernel)
    end = datetime.now()
    results.append(convolved_image)
tuner.observe(token, reward = start - end)
```

(b) Cuttlefish tuner to adaptively convolve the images

From:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
ML Techniques for Implementing DB Internals

- How does it learn?
  - _Cuttlefish_ uses contextual bandits, which is a simplified form of RL
  - It also employs tunable exploration policies

<table>
<thead>
<tr>
<th>Learn model of outcomes</th>
<th>Multi-armed bandits</th>
<th>Reinforcement Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given model of stochastic outcomes</td>
<td>Decision theory</td>
<td>Markov Decision Process</td>
</tr>
</tbody>
</table>

| Actions don’t change state of the world | Actions change state of the world |

Table 1: Four scenarios when reasoning under uncertainty.\(^1\)

From:
Kaftan et. al, 2018, _Cuttlefish: A Lightweight Primitive for Adaptive Query Processing_.
Zhou et al, 2016, _A survey on contextual multi-arm bandits_
ML Techniques for Implementing DB Internals

From:
ML Techniques for Implementing DB Internals

Algorithm 3 General Framework of Thompson Sampling

\[
\text{Define } D = \{\}\n\text{for } t = 1, \ldots, T \text{ do }
\begin{align*}
&\text{Receive context } x_t \\
&\text{Draw } \theta_t \text{ from posterior distribution } P(\theta|D) \\
&\text{Select arm } a_t = \arg \max_a E(r|x_t, a, \theta_t) \\
&\text{Receive reward } r_t \\
&D = D \cup \{x_t, a_t, r_t\}
\end{align*}
\text{end for}
\]

○ Given:
  - \(D\) past observations
  - \(x\) as context, \(a\) as arm/action, \(r\) as reward
  - Theta is a parameter (e.g. the mean) of the distribution learned for each arm

▷ Other variants for bandits exist and could be applicable.

From:
Zhou et al, 2016, A survey on contextual multi-arm bandits
ML Techniques for Implementing DB Internals

![Convolution relative throughput]

**Figure 9: Convolution relative throughput**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Physical Operators</th>
<th>Dataset</th>
<th>Workload Variants</th>
<th>Tuning Rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>3 Convolution Algs.</td>
<td>8091 Flickr Images (32 GB)</td>
<td>3 Stationary, 3 Dynamic</td>
<td>8091 (one per image)</td>
</tr>
<tr>
<td>Regex Matching</td>
<td>4 Regex Libraries</td>
<td>255,985 Common Crawl Docs (29.7 GB)</td>
<td>8 Regexes</td>
<td>255,985 (one per doc)</td>
</tr>
<tr>
<td>Parallel Join</td>
<td>2 Local Join Algs.</td>
<td>TPC-DS Data (about 200 GB)</td>
<td>23 Queries</td>
<td>512 (one per partition)</td>
</tr>
<tr>
<td>Simulated</td>
<td>2 to 50</td>
<td>—</td>
<td>14 Simulation Configs.</td>
<td>50000</td>
</tr>
</tbody>
</table>

▷ Generally good results so far.
▷ Context does not work all the time.

From:
ML Techniques for Implementing DB Internals

- Database Learning: Answering to queries with models

**Figure 1:** An example of how database learning might continuously refine its model as more queries are processed: after processing (a) 2 queries, (b) 4 queries, and (c) 8 queries. We could deliver more accurate answers if we combined this model with the approximate answers produced by traditional sampling techniques.

From:
Park, et. al., 2018. Database Learning: Towards a database that becomes smarter every time. SIGMOD
Self-managing databases

- Relational databases work on the principle of a declarative interface, such that underneath automatic optimizations can take place.
- From a high-level perspective these automated optimizations can take place, in either:
  - The storage engine
  - The query engine
  - The external management
- For adapting to workloads and devices, a large part of them is configured by Database Administrators (DBAs)
Self-managing databases

- DBAs can make mistakes, and are a costly component in the system.
- Human talent could also be employed elsewhere

From:
Self-managing databases

▷ Self-managing components in databases could alleviate this situation
Self-managing databases

- Hence this has been a grand goal of the research community for years

### TCDE Workgroup on Self-Managing Database Systems

#### Previous Workshops

<table>
<thead>
<tr>
<th>Year</th>
<th>Workshop Chairs</th>
<th>Keynote Speaker</th>
<th>Website</th>
<th>Workshop Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>Alexandros Labrinidis (University of Pittsburgh) Florian Waas (EMC Greenplum)</td>
<td>Timos Sellis (RMIT University)</td>
<td>Original</td>
<td>-</td>
</tr>
<tr>
<td>2012</td>
<td>Alejandro (Alex) Buchmann (Technische Universitaet Darmstadt) Malu Castellanos (HP Labs)</td>
<td>Nachum Shacham (eBay)</td>
<td>Original</td>
<td>-</td>
</tr>
<tr>
<td>2011</td>
<td>Vivek Narasayya (Microsoft Research) Neoclis Polyzotis (University of California - Santa Cruz)</td>
<td>Rob Woollen (Salesforce.com)</td>
<td>Original</td>
<td>-</td>
</tr>
<tr>
<td>2010</td>
<td>Shiniath Babu (Duke University) Kai-Uwe Sattler (Technische Universitat Ilmenau)</td>
<td>Oliver Ratzesberger (eBay)</td>
<td>Original</td>
<td>Report in DEB</td>
</tr>
<tr>
<td>2009</td>
<td>Ashraf Aboulnaga (University of Waterloo) Ken Salem (University of Waterloo)</td>
<td>James Hamilton (Amazon Web Services)</td>
<td>Original</td>
<td>Report in DEB</td>
</tr>
<tr>
<td>2008</td>
<td>Pat Martin (Queen’s University)</td>
<td>John Wilkes (HP Labs)</td>
<td>Local Mirror</td>
<td>Report in DEB</td>
</tr>
<tr>
<td>2007</td>
<td>Guy Lohman (IBM Almaden Research Center)</td>
<td>-</td>
<td>Local Mirror</td>
<td>Report in DEB</td>
</tr>
<tr>
<td>2005</td>
<td>Takeshi Fukuda (IBM Yamato Software Laboratory)</td>
<td>Guy Lohman (IBM Almaden Research Center)</td>
<td>Original</td>
<td>-</td>
</tr>
</tbody>
</table>

From:
http://dsg.uwaterloo.ca/tcde-smdb/#workshops
Self-managing databases

Through ongoing literature research I would like to understand better the state of the development:

What are the grand goals, what is missing?

Is there sufficient follow-up on innovative research?

Do we really have self-driving DBMSs?

From:
Self-Managing Operational Aspects

- There are ongoing developments that study the self-management of operational aspects
  - Configuration knobs (e.g. Ottertune)
  - Forecasting-based provisioning (e.g. P-store)
- Limited research papers from the Oracle team
Self-Managing Operational Aspects

Oracle. A very fuzzy example

 Applying Machine Learning to Database Faults

From: http://idcdocserv.com/US43571317
Self-Managing Operational Aspects

▷ OtterTune (2017)
  ○ Motivation:
    ■ Difficult to find an optimal configuration for several workloads

From:
Self-Managing Operational Aspects

OtterTune

Motivation:
- Number of knobs grow, each has a different distribution for impact on performance, and there’s interdependence

From:
**Self-Managing Operational Aspects**

- **OtterTune**
  - Factor analysis for correlation understanding, K-means for clustering metrics, Lasso for knob ranking.
  - Gaussian process for exploration (similar to bandit)

Self-Managing Operational Aspects

▶ OtterTune

From:
Self-Managing Operational Aspects

▷ P-Store (2018)
  ○ Motivation:
    ■ Skew and workload variation for elastic scaling in OLTP systems

From:
Self-Managing Operational Aspects

▷ P-Store
  ○ Motivation:
    ■ Reactive scaling causes unacceptable latency spikes

From:
Self-Managing Operational Aspects

▷ P-Store
  ○ Their solution: Time-series prediction and dynamic programming solution to determine actions.
  ○ They outperform reactive and static allocation strategies.

From:
Self-Managing Operational Aspects

▷ P-Store

Results – Static, Peak Provisioning
Machines Used: 10

From:
Self-Managing Operational Aspects

▷ P-Store

Results – Static, Average Provisioning
Machines Used: 4

Self-Managing Operational Aspects

▷ P-Store

Results – Reactive Scaling
Avg. Machines Used: 4.02

From:
Self-Managing Operational Aspects

▷ P-Store: Best solution => Less machines used, Lower Latency, Almost no reconfiguration penalty on latency

Results – P-Store with SPAR
Avg. Machines Used: 5.05

From:
Self-Managing Operational Aspects

- Time-series are useful for reasoning about workload changes!
Self-Managing Database Internals

▷ A long history in finding knobs (e.g. index selection, different layouts, partitioning, variant selection, access-path selection) specific to given DBMS designs.
▷ Few unifying views on making this choice self-managing and connecting good ideas.
▷ As a good example from the area I will simply introduce Peloton.
  ○ Closest thing we’ve got to a unifying view nowadays.
Self-Managing Database Internals

▷ Peloton (2017)


Limited progress on this
Peloton: Forecasting works better on logical features, Ensembles needed, Many open research questions

From:
Ma, et. al., 2018. Query-based Workload Forecasting for Self-Driving Database Management Systems. SIGMOD
Peloton: Forecasting works better on logical features, Ensembles needed, Many open research questions

From:
Ma, et. al., 2018. Query-based Workload Forecasting for Self-Driving Database Management Systems. SIGMOD
Self-Managing Database Internals

The community is interested: No one has all the answers!
Reinforcement Learning is being widely studied

Storage Engine:

○ Index selection

○ Data Partitioning
Reinforcement Learning is being widely studied.

Self-Managing Database Internals

▷ Query Engine
  ○ Join Order Enumeration
  ○ Operator Variant Selection
  ○ Query Plan Optimization
3. A deeper look into a case: Learned Index Structures
Indexes are essential for efficient data access

Range queries (B-Trees), key lookups (Hash Maps), existence queries (Bloom Filters) rely on tuned index structures

- These indexes are general purpose and don’t fit for the worst-case distribution of the data.

From:
Kraska, et. al, 2018. The case for learned index structures. SIGMOD.
Thanks to Taranpreet Kaur for sharing material for these slides
The Case for Learned Index Structures

▷ The core idea of the paper:
  ○ All index structures can be replaced with other type of models, the authors call these LEARNED INDEXES.
  ○ They can be neural nets, or linear regressors, etc.
  ○ This reconstructs query results with low cost, acting as a form of compression.

From: Kraska, et. al, 2018. The case for learned index structures. SIGMOD.
The Case for Learned Index Structures

Forward-propagation or inference in basic NNs

1. The input is multiplied by the matrix connecting the input to the first hidden layer.
2. Bias is added (if present)
3. An activation function is applied e.g Relu (everything <0 is removed) or Sigmoid
4. The result is the input to the next layer...
5. This is repeated till the final layer:
A series of small matrix operations, with multiplication being the most heavy-weight

From:
The Case for Learned Index Structures

- Two general expectations when comparing index search with inference:

<table>
<thead>
<tr>
<th>Index Search</th>
<th>NN Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Whole data</td>
<td>+Reduced data used</td>
</tr>
<tr>
<td>-Sequential</td>
<td>+More parallelism available per search (since matrix multiplications are embarrassingly parallel)</td>
</tr>
</tbody>
</table>
Recursive Model Index: hierarchy of models, where at each stage the model takes the key as an input and based on it picks another model, until the final stage predicts the position. End result will be a page that can answer the lookup query.

Benefits: Easy to learn the overall data distribution, divides spaces into smaller sub ranges (like B-Trees), no search process required between stages. Hybridity.

- Inner nodes are classifiers
- Leaves are linear regressors

From: Kraska, et. al, 2018. The case for learned index structures. SIGMOD.
This model starts of by training the top nodes first.

It then predicts the next models in line 9 and 10 and stores all the keys that fall into that particular model (for training it later).

Also in case of hybrid index they calculate the error in line 12. If the error is higher than the threshold, the model is substituted by a B-tree.

Hybrid indexes bound the worst case performance of learned indexes to the performance of B-Trees.

This is their selling point:
- a “no tradeoffs” operation.

From: Kraska, et. al, 2018. The case for learned index structures. SIGMOD.
The Case for Learned Index Structures

Results for B-trees

<table>
<thead>
<tr>
<th>Type</th>
<th>Config</th>
<th>Map Data</th>
<th></th>
<th>Web Data</th>
<th></th>
<th>Log-Normal Data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Size (MB)</td>
<td>Lookup (ns)</td>
<td>Model (ns)</td>
<td>Size (MB)</td>
<td>Lookup (ns)</td>
<td>Model (ns)</td>
</tr>
<tr>
<td>Btree</td>
<td>page size: 32</td>
<td>52.45 (4.00x)</td>
<td>274 (0.97x)</td>
<td>198 (72.3%)</td>
<td>51.93 (4.00x)</td>
<td>276 (0.94x)</td>
<td>201 (72.7%)</td>
</tr>
<tr>
<td></td>
<td>page size: 64</td>
<td>26.23 (2.00x)</td>
<td>277 (0.96x)</td>
<td>172 (62.0%)</td>
<td>25.97 (2.00x)</td>
<td>274 (0.95x)</td>
<td>171 (62.4%)</td>
</tr>
<tr>
<td></td>
<td>page size: 128</td>
<td>13.11 (1.00x)</td>
<td>265 (1.00x)</td>
<td>134 (50.8%)</td>
<td>12.98 (1.00x)</td>
<td>260 (1.00x)</td>
<td>132 (50.8%)</td>
</tr>
<tr>
<td></td>
<td>page size: 256</td>
<td>6.56 (0.50x)</td>
<td>267 (0.99x)</td>
<td>114 (42.7%)</td>
<td>6.49 (0.50x)</td>
<td>266 (0.98x)</td>
<td>114 (42.9%)</td>
</tr>
<tr>
<td></td>
<td>page size: 512</td>
<td>3.28 (0.25x)</td>
<td>286 (0.93x)</td>
<td>101 (35.3%)</td>
<td>3.25 (0.25x)</td>
<td>291 (0.89x)</td>
<td>100 (34.3%)</td>
</tr>
<tr>
<td>Learned Index</td>
<td>2nd stage models: 10k</td>
<td>0.15 (0.01x)</td>
<td>98 (2.70x)</td>
<td>31 (31.6%)</td>
<td>0.15 (0.01x)</td>
<td>222 (1.17x)</td>
<td>29 (13.1%)</td>
</tr>
<tr>
<td></td>
<td>2nd stage models: 50k</td>
<td>0.76 (0.06x)</td>
<td>85 (3.11x)</td>
<td>39 (45.9%)</td>
<td>0.76 (0.06x)</td>
<td>162 (1.60x)</td>
<td>36 (22.2%)</td>
</tr>
<tr>
<td></td>
<td>2nd stage models: 100k</td>
<td>1.53 (0.12x)</td>
<td>82 (3.21x)</td>
<td>41 (50.2%)</td>
<td>1.53 (0.12x)</td>
<td>144 (1.81x)</td>
<td>39 (26.9%)</td>
</tr>
<tr>
<td></td>
<td>2nd stage models: 200k</td>
<td>3.05 (0.23x)</td>
<td>86 (3.08x)</td>
<td>50 (58.1%)</td>
<td>3.05 (0.24x)</td>
<td>126 (2.07x)</td>
<td>41 (32.5%)</td>
</tr>
</tbody>
</table>

From:
Kraska, et. al, 2018. The case for learned index structures. SIGMOD.
The Case for Learned Index Structures

Our Results for B-trees

From:
Results from Master Thesis of Taranpreet Kaur: Learned Index Structures: Practical Implementations and Future Directions
4.

Our work in the area

(an invitation to collaborate in the near future)
Project Research Interests

**Action Planning & Engineering**
- RL for automated physical design
- RL for query processing
- Work on horizontal partitioning
- Concepts for fair work distribution

**Software 2.0 for Database Internals**
- Learned index structures,
- Multi-dimensional learned index structures

**Workload Forecasting**
- Models for reasoning about workloads
- HTAP Workload Benchmark
- Time-series motif analysis
- Time-series clustering and forecasting

**DB Support for some ML tasks**
- Enabling skipping for word embeddings in parquet-like horizontal partitions (for ER)
- Automated model management and tracking
- Explainability

**Co-Processor Acceleration and Hardware Tuning**
- GPU-powered HTAP design
- Code generation for operators
- Real-world cases for HTAP

**Non-Relational Models**
- Model changes as an automated physical design knob
- Support for approximate graph representations
- Graph workload adaptivity, for dynamic graphs based on Scholarly NA
Thanks for the attention!

Questions? Interested in collaborating? Ideas?

You can find me at:
campero@ovgu.de
G29-145
References


References


Recommended Resources

▷ https://a16z.com/2016/06/10/ai-deep-learning-machines/
▷ https://cloud.withgoogle.com/build/data-analytics/explore-history-machine-learning/
▷ https://developers.google.com/machine-learning/crash-course/ml-intro