Learned Algorithms and Data Structures

Tim Kraska <kraska@mit.edu>

[Disclaimer: I am NOT talking on behalf of Google]
Learned Algorithms and Data Structures

“Machine Learning Just Ate Algorithms In One Large Bite, thx to @tim_kraska, @alexbeutel, @edchi, @JeffDean & Polyzotis, …” [Christopher Manning, Professor at Stanford]
Learned Algorithms and Data Structures

Tim Kraska
<kraska@mit.edu>

“Machine Learning Just Ate Algorithms In One Large Bite, thx to @tim_kraska, @alexbeutel, @edchi, @JeffDean & Polyzotis, …” [Christopher Manning, Professor at Stanford]
Fundamental Building Blocks

- Sorting
- B-Tree
- Hash-Map
- Scheduling
- Priority Queue
- Bloom Filter
- Range Filter
- Caching
- Join
Used whenever efficient data access is needed from database systems to mobile applications
Key (e.g., book title, author,...)
Key
(e.g., book title, author,...)

A-B B-C C-G

AA- AL AK AP

BA- BE BL BR

...
Key
(e.g., book title, author, ...)

[Diagram of a tree structure with labels A, B, C, G, and further branching with AL, AM, AN, AP, etc.]
Model predicts the location of the data like the librarian predicts the location of the book.
Not convinced yet?
Another Example:

Index All Integers from 900 to 800M

B-Tree?
A More Concrete Example:

Index All Integers from 900 to 800M

900 901 902 903 904 905 906 907 908 909 ... 800M

data_array[lookup_key - 900]
Goal:

Index All Integers from 900 to 800M

Index All Even Integers from 900 to 800M

\[
data\_array[(lookup\_key - 900) / 2]
\]
Conceptually a B-Tree maps a key to a page.

For simplicity assume all pages are continuously stored in main memory.
Alternative View

B-Tree maps a key to a position with a fixed min/max error

For simplicity assume all pages are continuously stored in main memory
A B-Tree Is A Model
A B-Tree Is A Model

Finding an item
1. Any model: key $\rightarrow$ pos estimate
2. Binary search in $[\text{pos} - \text{err}_{\text{min}}, \text{pos} + \text{err}_{\text{max}}]$

$\text{err}_{\text{min}}$ and $\text{err}_{\text{max}}$ are known from the training process.
A B-Tree Is A Model

A form of a regression model

key → pos is equivalent of modeling the CDF of the (observed) key distribution:
Pos-estimate = \( P(X \leq \text{key}) \times \#\text{keys} \)
A B-Tree Is A Model

Pos-estimate = $F(key) \times \#keys$
B-Trees Are Regression Trees

![Diagram](image-url)
What Does This Mean
Database people were the first to do large scale machine learning :)

What Does This Mean
Why Is This A Big Deal?
<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>first_name</th>
<th>last_name</th>
<th>email</th>
<th>address</th>
<th>zip</th>
<th>state</th>
<th>credit_card_nb</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2017-01-01</td>
<td>Hobart</td>
<td>Barnaby</td>
<td><a href="mailto:bbarnaby13@goo.ne.jp">bbarnaby13@goo.ne.jp</a></td>
<td>20565 High Crossing Plaza</td>
<td>56372</td>
<td>Minnesota</td>
<td>4405-6979-7285-5160</td>
<td>$611.00</td>
</tr>
<tr>
<td>1001</td>
<td>2017-01-02</td>
<td>Billye</td>
<td>Binness</td>
<td><a href="mailto:bnussen1r@topsy.com">bnussen1r@topsy.com</a></td>
<td>3698 Upham Point</td>
<td>20260</td>
<td>District of Columbia</td>
<td>3533-7150-7728-9850</td>
<td>$244.00</td>
</tr>
<tr>
<td>1002</td>
<td>2017-01-02</td>
<td>Johann</td>
<td>Brockley</td>
<td><a href="mailto:jbrockley2@bimjournals.com">jbrockley2@bimjournals.com</a></td>
<td>23844 Artisan Place</td>
<td>98516</td>
<td>Washington</td>
<td>67597-1193-7985-5100</td>
<td>$233.00</td>
</tr>
<tr>
<td>1003</td>
<td>2017-01-03</td>
<td>Arbie</td>
<td>MacMenami</td>
<td><a href="mailto:amacmenami3@hao123.com">amacmenami3@hao123.com</a></td>
<td>6276 Toban Trail</td>
<td>78759</td>
<td>Texas</td>
<td>3537-4829-6134-5000</td>
<td>$210.00</td>
</tr>
<tr>
<td>1004</td>
<td>2017-01-03</td>
<td>Delilah</td>
<td>O’Currgan</td>
<td><a href="mailto:docurrgan4@chron.com">docurrgan4@chron.com</a></td>
<td>86016 New Castle Avenue</td>
<td>72199</td>
<td>Arkansas</td>
<td>3555-2017-2226-5780</td>
<td>$286.00</td>
</tr>
<tr>
<td>1005</td>
<td>2017-01-04</td>
<td>Greta</td>
<td>Will</td>
<td><a href="mailto:gwill5@yelp.com">gwill5@yelp.com</a></td>
<td>0 Dottie Circle</td>
<td>68524</td>
<td>Nebraska</td>
<td>503844-1984-2065-5000</td>
<td>$870.00</td>
</tr>
<tr>
<td>1006</td>
<td>2017-01-04</td>
<td>Gordon</td>
<td>Kirsopp</td>
<td><a href="mailto:gkirsopp6@utexas.edu">gkirsopp6@utexas.edu</a></td>
<td>64060 Scott Park</td>
<td>20370</td>
<td>District of Columbia</td>
<td>633332-1895-2414-5000</td>
<td>$687.00</td>
</tr>
<tr>
<td>1007</td>
<td>2017-01-05</td>
<td>Bendick</td>
<td>Fagg</td>
<td><a href="mailto:bfgagg7@army.mil">bfgagg7@army.mil</a></td>
<td>94 Florence Hill</td>
<td>45440</td>
<td>Ohio</td>
<td>3528-9673-1815-8420</td>
<td>$733.00</td>
</tr>
<tr>
<td>1008</td>
<td>2017-01-05</td>
<td>Dimitry</td>
<td>Boyet</td>
<td><a href="mailto:dboyet8@sakura.ne.jp">dboyet8@sakura.ne.jp</a></td>
<td>35886 Golf Plaza</td>
<td>30066</td>
<td>Georgia</td>
<td>3576-6991-4041-3170</td>
<td>$382.00</td>
</tr>
<tr>
<td>1009</td>
<td>2017-01-06</td>
<td>Ailsun</td>
<td>Beinke</td>
<td><a href="mailto:aabinkel9@si.edu">aabinkel9@si.edu</a></td>
<td>1 Badeau Place</td>
<td>46295</td>
<td>Indiana</td>
<td>56022-2011-8072-1400</td>
<td>$854.00</td>
</tr>
<tr>
<td>1010</td>
<td>2017-01-07</td>
<td>Lou</td>
<td>Hallows</td>
<td><a href="mailto:lhallowsa@theguardian.com">lhallowsa@theguardian.com</a></td>
<td>1 Twin Pines Junction</td>
<td>91125</td>
<td>California</td>
<td>5602-2364-4079-0250</td>
<td>$150.00</td>
</tr>
<tr>
<td>1011</td>
<td>2017-01-09</td>
<td>Tiffani</td>
<td>Mathew</td>
<td><a href="mailto:tmathewb@seattletimes.com">tmathewb@seattletimes.com</a></td>
<td>0456 Meadow Vale Lane</td>
<td>75260</td>
<td>Texas</td>
<td>6387-6943-8910-4580</td>
<td>$313.00</td>
</tr>
<tr>
<td>1012</td>
<td>2017-01-09</td>
<td>Perl</td>
<td>Bridie</td>
<td><a href="mailto:pbridiec@hubpages.com">pbridiec@hubpages.com</a></td>
<td>07 Bluestem Junction</td>
<td>33124</td>
<td>Florida</td>
<td>3539-8662-2397-5880</td>
<td>$558.00</td>
</tr>
<tr>
<td>1013</td>
<td>2017-01-09</td>
<td>Rosabelle</td>
<td>Blasik</td>
<td><a href="mailto:rblasik8@delicious.com">rblasik8@delicious.com</a></td>
<td>7 Fairfield Pass</td>
<td>79699</td>
<td>Texas</td>
<td>5662-2297-6599-8560</td>
<td>$941.00</td>
</tr>
<tr>
<td>1014</td>
<td>2017-01-10</td>
<td>Meggi</td>
<td>Belamy</td>
<td><a href="mailto:mbelamye@ask.com">mbelamye@ask.com</a></td>
<td>0995 Manufacturers Street</td>
<td>10170</td>
<td>New York</td>
<td>3557-5094-7405-8340</td>
<td>$875.00</td>
</tr>
<tr>
<td>1015</td>
<td>2017-01-10</td>
<td>Tadio</td>
<td>Balderson</td>
<td><a href="mailto:tbalderson@apache.org">tbalderson@apache.org</a></td>
<td>80 Novick Road</td>
<td>75260</td>
<td>Texas</td>
<td>60485-7328-7199-9300</td>
<td>$954.00</td>
</tr>
<tr>
<td>1016</td>
<td>2017-01-11</td>
<td>Gianina</td>
<td>Ostbye</td>
<td><a href="mailto:goostbyeq@google.pl">goostbyeq@google.pl</a></td>
<td>72674 Fuller Avenue</td>
<td>89505</td>
<td>Nevada</td>
<td>4-0415-9268-2397</td>
<td>$239.00</td>
</tr>
<tr>
<td>1017</td>
<td>2017-01-12</td>
<td>Brendan</td>
<td>Doody</td>
<td><a href="mailto:bdoodyh@cragslist.org">bdoodyh@cragslist.org</a></td>
<td>87414 Golden Leaf Street</td>
<td>11480</td>
<td>New York</td>
<td>206-6348-4121-1314</td>
<td>$308.00</td>
</tr>
<tr>
<td>1018</td>
<td>2017-01-13</td>
<td>Conway</td>
<td>Coombs</td>
<td><a href="mailto:ccombs@blogger.com">ccombs@blogger.com</a></td>
<td>2810 Oakridge Park</td>
<td>32859</td>
<td>Florida</td>
<td>3529-1514-0357-9120</td>
<td>$60.00</td>
</tr>
<tr>
<td>1019</td>
<td>2017-01-14</td>
<td>Germaine</td>
<td>Bere</td>
<td><a href="mailto:gberej@bravesites.com">gberej@bravesites.com</a></td>
<td>82802 Oakridge Park</td>
<td>20041</td>
<td>District of Columbia</td>
<td>670961-0240-4054-9000</td>
<td>$95.00</td>
</tr>
<tr>
<td>1020</td>
<td>2017-01-15</td>
<td>Davide</td>
<td>Tolcharde</td>
<td><a href="mailto:dtolchardek@redcross.org">dtolchardek@redcross.org</a></td>
<td>89 Continental Avenue</td>
<td>79165</td>
<td>Texas</td>
<td>5018-7748-4325-9510</td>
<td>$137.00</td>
</tr>
<tr>
<td>1021</td>
<td>2017-01-16</td>
<td>Nigel</td>
<td>Artharg</td>
<td><a href="mailto:narthargl@gizmodo.com">narthargl@gizmodo.com</a></td>
<td>31 McBride Point</td>
<td>22301</td>
<td>Virginia</td>
<td>560225-6965-2870-0000</td>
<td>$496.00</td>
</tr>
<tr>
<td>1022</td>
<td>2017-01-17</td>
<td>Richard</td>
<td>Trenholm</td>
<td><a href="mailto:rtrenholm@csblocal.com">rtrenholm@csblocal.com</a></td>
<td>93 Hoepek Parkway</td>
<td>70593</td>
<td>Louisiana</td>
<td>3541-5241-5383-9970</td>
<td>$760.00</td>
</tr>
<tr>
<td>1023</td>
<td>2017-01-18</td>
<td>Juditha</td>
<td>Dwane</td>
<td><a href="mailto:jdwanen@vk.com">jdwanen@vk.com</a></td>
<td>7914 Eliot Lane</td>
<td>44276</td>
<td>Texas</td>
<td>5456-4410-0914-3180</td>
<td>$474.00</td>
</tr>
<tr>
<td>1024</td>
<td>2017-01-19</td>
<td>Susan</td>
<td>Iden</td>
<td><a href="mailto:silden@ao.com">silden@ao.com</a></td>
<td>25204 Huxley Road</td>
<td>21684</td>
<td>Maryland</td>
<td>3574-8586-6367-9920</td>
<td>$83.00</td>
</tr>
<tr>
<td>1025</td>
<td>2017-01-20</td>
<td>Abbey</td>
<td>Triggle</td>
<td><a href="mailto:atrigglep@google.com.au">atrigglep@google.com.au</a></td>
<td>47 Debra Pass</td>
<td>74184</td>
<td>Oklahoma</td>
<td>3538-6047-6315-7710</td>
<td>$513.00</td>
</tr>
<tr>
<td>1026</td>
<td>2017-01-21</td>
<td>Zsazsa</td>
<td>Dunster</td>
<td><a href="mailto:zdunsterg@nature.com">zdunsterg@nature.com</a></td>
<td>7 Gerald Alley</td>
<td>40576</td>
<td>Kentucky</td>
<td>3562-0325-7709-3490</td>
<td>$952.00</td>
</tr>
</tbody>
</table>
Adaptation To Tenant’s Data

data_array[id - 1000]

data_array[model(date)]
Adaptation To Tenant’s Data and Workload
System Customization Through Models

Building a system from scratch for every use case is not economical.

Machine Learning makes it possible and we can leverage decades of ML research.
Why Is This A Big Deal?

Adapts to data and workload

More efficient for CPU and memory

GPUs/FPGAs/TPUs

Cheaper inserts
Instance Optimality

Instance-optimality defines that an algorithm $b$ is instance optimal over a class of algorithm $A$ and a database $d$, if

$$
cost(b,d) \leq c \cdot cost(a,d) + c' \quad \forall \ a \in A
$$
Does It Work? A First Attempt

State-Of-The-Art B-Tree

260ns

TensorFlow

???
Does It Work? A First Attempt

State-Of-The-Art B-Tree

260ns

TensorFlow

>80,000ns
Challenges

Traditional model architectures do not work

Frameworks are not designed for nano-second execution

Overfitting can be good

ML+System Co-Design

underfitting  desired  overfitting  desired
Problem I: The Learning Index Framework (LIF)

• An index synthesis system

• Given an index configuration generate the best possible code

• Uses ideas from Tupleware [VLDB15]

• Simple models are trained “on-the-fly”, whereas for complex models we use Tensorflow and extract weights afterwards (i.e., no Tensorflow during inference time)

• Best index configuration is found using auto-tuning (e.g., see TuPAQ [SOCC15])
Problem II + III: Precision Gain per Node

Index over 100M records. Page-size: 100

Precision Gain: 100M --> 1M
(Min/Max-Error: 1M)

Precision Gain: 1M --> 10k

Precision Gain: 10k --> 100

100M records
(i.e., 1M pages)
The Last Mile Problem
Solution:
Recursive Model Index (RMI)

\[ L_0 = \sum_{(x,y)} (f_0(x) - y)^2 \]

\[ L_\ell = \sum_{(x,y)} (f_\ell(M_\ell f_{\ell-1}(x)/N))(x) - y)^2 \]
How Does The Lookup-Code Look Like

Model on stage 1: \( f_0(\text{key\_type } \text{key}) \)

Models on stage two: \( f_1[] \) (e.g., the first model in the second stage is \( f_1[0](\text{key\_type } \text{key}) \))

Lookup Code for a 2-stage RMI:

```plaintext
    pos_estimate <- f1[f0(key)](key)
    pos <- exp_search(key, pos_estimate, data);
```
How Does The Lookup-Code Look Like

Model on stage 1: \( f_0(\text{key\_type \ key}) \)

Models on stage two: \( f_1[] \) (e.g., the first model in the second stage is \( f_1[0](\text{key\_type \ key}) \))

Lookup Code for a 2-stage RMI:

\[
\begin{align*}
\text{pos\_estimate} & \leftarrow f_1[f_0(\text{key})](\text{key}) \\
\text{pos} & \leftarrow \text{exp\_search(}\text{key, pos\_estimate, data)};
\end{align*}
\]

Operations with a 2-stage RMI with linear regression models

\[
\begin{align*}
\text{offset} & \leftarrow a + b \times \text{key} \\
\text{weights2} & \leftarrow \text{weights\_stage2}[\text{offset}] \\
\text{pos\_estimate} & \leftarrow \text{weights2.a} + \text{weights2.b} \times \text{key} \\
\text{pos} & \leftarrow \text{exp\_search(}\text{key, pos\_estimate, data)};
\end{align*}
\]

2x multiplies
2x additions
1x array-lookup
Worst-Case Performance is the one of a B-Tree
Problem: Min-/Max-Error vs Average Error
Binary Search
Binary Search
Binary Search

Predicted Position

Actual Position

0 ........................ N

Left  Middle  Right
Quaternary Search
Quaternary Search

- Predicted Position
  - Q1: Prediction – 2x std err
  - Q2
  - Q3: Prediction + 2x std err

- Actual Position
- Left
- Right
Quaternary Search

Predicted Position

Actual Position

Left Q1 Q2 Q3 Right

0 N
Exponential Search

Predicted Position

Actual Position
Does it have to be

DEEP LEARNING
Initial Results

TensorFlow

> 80,000ns

State-Of-The-Art

B-Tree

265ns

13MB

Learned Index

85ns

0.7MB
Does It Work?

200M records of map data (e.g., restaurant locations). Index on longitude.
Intel-E5 CPU with 32GB RAM without GPU/TPUs No Special SIMD optimization (there is a lot of potential)

<table>
<thead>
<tr>
<th>Type</th>
<th>Config</th>
<th>Lookup time</th>
<th>Speedup vs. BTree</th>
<th>Size (MB)</th>
<th>Size vs. Btree</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTree</td>
<td>page size: 128</td>
<td>260 ns</td>
<td>1.0X</td>
<td>12.98 MB</td>
<td>1.0X</td>
</tr>
<tr>
<td>Learned index</td>
<td>2nd stage size: 10000</td>
<td>222 ns</td>
<td>1.17X</td>
<td>0.15 MB</td>
<td>0.01X</td>
</tr>
<tr>
<td>Learned index</td>
<td>2nd stage size: 50000</td>
<td>162 ns</td>
<td>1.60X</td>
<td>0.76 MB</td>
<td>0.05X</td>
</tr>
<tr>
<td>Learned index</td>
<td>2nd stage size: 100000</td>
<td>144 ns</td>
<td>1.67X</td>
<td>1.53 MB</td>
<td>0.12X</td>
</tr>
<tr>
<td>Learned index</td>
<td>2nd stage size: 200000</td>
<td>126 ns</td>
<td>2.06X</td>
<td>3.05 MB</td>
<td>0.23X</td>
</tr>
</tbody>
</table>

60% faster at 1/20th the space, or 17% faster at 1/100th the space.
You Might Have Seen Certain Blog Posts
Initial Results

- **Learned Index**
- **Lookup Table**
- **FAST**
- **Fixed-Size Read-Optimized B-Tree w/ Interpolation Search**
A Comparison To ARTful Indexes (Radix-Tree)

Viktor Leis, Alfons Kemper, Thomas Neumann: The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases. ICDE 2013

Experimental setup:

• Dense: continuous keys from 0 to 256M
• Sparse: 256M keys where each bit is equally likely 0 or 1.
A Comparison To ARTful Indexes (Radix-Tree)

Viktor Leis, Alfons Kemper, Thomas Neumann: The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases. ICDE 2013

Experimental setup: continuous keys from 0 to 256M

Reported lookup throughput: $10\text{M/s} \approx 100\text{ns}^{(1)}$

Size: not measured, but paper says overhead of $\approx 8$ Bytes per key (dense, best case): $256\text{M} * 8 \text{ Byte} \approx 1953\text{MB}$

$^{(1)}$Numbers from the paper
Learned Index

Generate Code:

```c
Record lookup(key) {
    return data[0 + 1 * key];
}
```
Learned Index

Generate Code:

```cpp
Record lookup(key) {
    return data[key];
}
```
Learned Index

Generate Code:

```java
Record lookup(key) {
    return data[key];
}
```

Lookup Latency: 10ns (learned index) vs 100ns* (ARTfull)
or one-order-of-magnitude better

Space: 0MB vs 1953MB
Infinitely better :(
UNFAIR?
Other Criticism

• Inserts (in a second)
• What if my data / workload changes
• This only works for primary indexes
  • Key/Value stores
  • Working on global-optimization secondary indexes
What about Updates and Inserts?
What about Updates and Inserts?

Alex Galakatos, Michael Markovitch, Carsten Binnig, Rodrigo Fonseca, Tim Kraska:
A-Tree: A Bounded Approximate Index Structure
To appear at SIGMOD 2019
The Simple Approach: Delta Indexing

Training a simple Multi-Variate Regression Model Can be done in one pass over the data
Leverage the Distribution for Appends

If the Learned Model Can Generalize to Inserts
Insert complexity is $O(1)$ not $O(\log N)$
Updates/Inserts

• Less beneficial as the data still has to be stored sorted
• Idea: Leave space in the array where more updates/inserts are expected
• Can also be done with traditional trees.
• But, the error of learned indexes should increase with $\sqrt{N}$ per node in RMI whereas traditional indexes with $N$
Still at the Beginning!

- Can we provide bounds for inserts?
- When to retrain?
- How to retrain models on the fly?
- …
Fundamental Algorithms & Data Structures

- Tree
- Hash-Map
- Bloom-Filter
- Multi-Dim Index
- Sorting
- Range-Filter
- DNA-Search

- Data Cubes
- Scheduling
- SQL Query Optimizer
- Cache Policy
- Join
- Nearest Neighbor
Fundamental Algorithms & Data Structures

Tree  Hash-Map  Bloom-Filter  Multi-Dim Index  Sorting  Range-Filter  DNA-Search

Data Cubes  Scheduling  SQL Query Optimizer  Cache Policy  Join  Nearest Neighbor  .....
Fundamental Algorithms & Data Structures

Our initial paper (CDF-based)
- Tree
- Hash-Map
- Bloom-Filter

Work in Progress (CDF-based)
- Multi-Dim Index
- Sorting
- Range-Filter
- DNA-Search
- CDF-Synth.

Work in Progress (Oracle/Full)
- Data Cubes
- Scheduling
- SQL Query Optimizer
- Cache Policy
- Join
- Nearest Neighbor

F(x)
4 Ways for ML-Enhanced Algorithms and Data Structures

- **Configure/synthesize** traditional algorithm using a model
- **CDF**: empirical CDF model of the data
- **Oracle**: prediction model
- **Full-Model**: learning the entire algorithm/data structure
Fundamental Algorithms & Data Structures

Our initial paper (CDF-based)
- Tree
- Hash-Map
- Bloom-Filter

Work in Progress (CDF-based)
- Multi-Dim Index
- Sorting
- Range-Filter
- DNA-Search
- CDF-Synth.
- \( F(x) \)

Work in Progress (Oracle/Full)
- Data Cubes
- Scheduling
- SQL Query Optimizer
- Cache Policy
- Join
- Nearest Neighbor
- …..
Fundamental Algorithms & Data Structures

Our initial paper (CDF-based)
- Tree
- Hash-Map
- Bloom-Filter

Work in Progress (CDF-based)
- Multi-Dim Index
- Sorting
- Range-Filter
- DNA-Search
- CDF-Synth.

Work in Progress (Oracle/Full)
- Data Cubes
- Scheduling
- SQL Query Optimizer
- Cache Policy
- Join
- Nearest Neighbor

\[ F(x) \]
Hash Map

Goal: Reduce Conflicts
## Hash Map – Example Results

<table>
<thead>
<tr>
<th>Type</th>
<th>Time (ns)</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford AVX Cuckoo, 4 Byte value</td>
<td>31ns</td>
<td>99%</td>
</tr>
<tr>
<td>Stanford AVX Cuckoo, 20 Byte record - Standard Hash</td>
<td>43ns</td>
<td>99%</td>
</tr>
<tr>
<td>Commercial Cuckoo, 20Byte record - Standard Hash</td>
<td>90ns</td>
<td>95%</td>
</tr>
<tr>
<td>In-place chained Hash-map, 20Byte record, <strong>learned hash functions</strong></td>
<td>35ns</td>
<td>100%</td>
</tr>
</tbody>
</table>
Fundamental Algorithms & Data Structures

Our initial paper (CDF-based)

- Tree
- Hash-Map
- Bloom-Filter

Work in Progress (CDF-based)

- Multi-Dim Index
- Sorting
- Range-Filter
- DNA-Search
- CDF-Synth.

Work in Progress (Oracle/Full)

- Data Cubes
- Scheduling
- SQL Query Optimizer

- Cache Policy
- Join
- Nearest Neighbor
### How To Build an Index For First and Last Name, Date, ID, and Email At The Same Time

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>first_name</th>
<th>last_name</th>
<th>email</th>
<th>address</th>
<th>zip</th>
<th>state</th>
<th>credit_card_nb</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>2017-01-01</td>
<td>Hobart</td>
<td>Spracklin</td>
<td><a href="mailto:hspracklin0@dailymotion.com">hspracklin0@dailymotion.com</a></td>
<td>20565 High Crossing Plaza</td>
<td>56372</td>
<td>Minnesota</td>
<td>4405-6975-7285-5160</td>
<td>$611.00</td>
</tr>
<tr>
<td>101</td>
<td>2017-01-02</td>
<td>Billye</td>
<td>Binnion</td>
<td>bbinnion@<a href="mailto:1@123-reg.co.uk">1@123-reg.co.uk</a></td>
<td>3698 Upham Point</td>
<td>20260</td>
<td>District of Columbia</td>
<td>3533-7150-7728-9850</td>
<td>$244.00</td>
</tr>
<tr>
<td>102</td>
<td>2017-01-02</td>
<td>Johann</td>
<td>Brockley2</td>
<td><a href="mailto:jbrockley2@bizjournals.com">jbrockley2@bizjournals.com</a></td>
<td>23844 Artisan Place</td>
<td>98516</td>
<td>Washington</td>
<td>67597-1193-7985-5100</td>
<td>$233.00</td>
</tr>
<tr>
<td>103</td>
<td>2017-01-03</td>
<td>Artie</td>
<td>MacMenami</td>
<td><a href="mailto:amacmenami3@hao123.com">amacmenami3@hao123.com</a></td>
<td>6276 Toban Trail</td>
<td>78759</td>
<td>Texas</td>
<td>3537-4829-6134-5000</td>
<td>$210.00</td>
</tr>
<tr>
<td>104</td>
<td>2017-01-03</td>
<td>Delilah</td>
<td>O’Currigan</td>
<td><a href="mailto:docurrigan4@chron.com">docurrigan4@chron.com</a></td>
<td>86016 New Castle Avenue</td>
<td>72199</td>
<td>Arkansas</td>
<td>3555-2017-2226-5780</td>
<td>$286.00</td>
</tr>
<tr>
<td>105</td>
<td>2017-01-04</td>
<td>Gretta</td>
<td>Will</td>
<td><a href="mailto:gwills@yelp.com">gwills@yelp.com</a></td>
<td>0 Dottie Circle</td>
<td>68524</td>
<td>Nebraska</td>
<td>503844-1984-2085-5000</td>
<td>$870.00</td>
</tr>
<tr>
<td>106</td>
<td>2017-01-04</td>
<td>Gordon</td>
<td>Kirssopp</td>
<td><a href="mailto:ggirssopp@rutaxas.edu">ggirssopp@rutaxas.edu</a></td>
<td>64060 Scott Park</td>
<td>20370</td>
<td>District of Columbia</td>
<td>633332-1895-2414-5000</td>
<td>$687.00</td>
</tr>
<tr>
<td>107</td>
<td>2017-01-05</td>
<td>Bendick</td>
<td>Fagg</td>
<td><a href="mailto:bfagg7@army.mil">bfagg7@army.mil</a></td>
<td>94 Florence Hill</td>
<td>45440</td>
<td>Ohio</td>
<td>3528-9673-1815-8420</td>
<td>$733.00</td>
</tr>
<tr>
<td>108</td>
<td>2017-01-05</td>
<td>Dimitry</td>
<td>Bayet</td>
<td><a href="mailto:dbayet8@sakura.ne.jp">dbayet8@sakura.ne.jp</a></td>
<td>35886 Golf Plaza</td>
<td>30066</td>
<td>Georgia</td>
<td>3576-6991-4041-3170</td>
<td>$382.00</td>
</tr>
<tr>
<td>109</td>
<td>2017-01-06</td>
<td>Ailsun</td>
<td>Beinke</td>
<td><a href="mailto:aibeinkek@si.edu">aibeinkek@si.edu</a></td>
<td>1 Badeau Place</td>
<td>46295</td>
<td>Indiana</td>
<td>56022-2011-8072-1400</td>
<td>$854.00</td>
</tr>
<tr>
<td>110</td>
<td>2017-01-07</td>
<td>Lou</td>
<td>Hallowsa</td>
<td>shhallowsa@the guardian.com</td>
<td>1 Twin Pines Junction</td>
<td>91125</td>
<td>California</td>
<td>5602-2364-4079-0250</td>
<td>$150.00</td>
</tr>
<tr>
<td>111</td>
<td>2017-01-07</td>
<td>Tiffani</td>
<td>Mathew</td>
<td><a href="mailto:tmathewb@seattletimes.com">tmathewb@seattletimes.com</a></td>
<td>0456 Meadow Vale Lane</td>
<td>75260</td>
<td>Texas</td>
<td>6387-6943-8910-4580</td>
<td>$313.00</td>
</tr>
<tr>
<td>112</td>
<td>2017-01-09</td>
<td>Peri</td>
<td>Bridie</td>
<td><a href="mailto:pbridiec@hubpages.com">pbridiec@hubpages.com</a></td>
<td>07 Bluestem Junction</td>
<td>33124</td>
<td>Nevada</td>
<td>3539-8662-2397-5880</td>
<td>$558.00</td>
</tr>
<tr>
<td>113</td>
<td>2017-01-09</td>
<td>Rosabelle</td>
<td>Blask</td>
<td><a href="mailto:rblasikd@delicious.com">rblasikd@delicious.com</a></td>
<td>7 Fairfield Pass</td>
<td>79699</td>
<td>Texas</td>
<td>5602-2297-6599-8560</td>
<td>$941.00</td>
</tr>
<tr>
<td>114</td>
<td>2017-01-10</td>
<td>Meggi</td>
<td>Belamy</td>
<td>mbelamy@<a href="mailto:j@ask.com">j@ask.com</a></td>
<td>0995 Manufacturers Street</td>
<td>10170</td>
<td>New York</td>
<td>3557-5094-7405-8340</td>
<td>$875.00</td>
</tr>
<tr>
<td>115</td>
<td>2017-01-10</td>
<td>Tadio</td>
<td>Balderston</td>
<td><a href="mailto:tbalderstonf@apache.org">tbalderstonf@apache.org</a></td>
<td>80 Novick Road</td>
<td>75260</td>
<td>Texas</td>
<td>60485-3728-7119-9300</td>
<td>$954.00</td>
</tr>
<tr>
<td>116</td>
<td>2017-01-11</td>
<td>Gianina</td>
<td>Otxbey</td>
<td><a href="mailto:gtxbeygb@ask.google.pl">gtxbeygb@ask.google.pl</a></td>
<td>72674 Fuller Avenue</td>
<td>89505</td>
<td>Nevada</td>
<td>4-0415-9268-2397</td>
<td>$239.00</td>
</tr>
<tr>
<td>117</td>
<td>2017-01-12</td>
<td>Brendan</td>
<td>Doody</td>
<td><a href="mailto:bdoodyh@crailslist.org">bdoodyh@crailslist.org</a></td>
<td>87414 Golden Leaf Street</td>
<td>11480</td>
<td>New York</td>
<td>201-6348-4121-1314</td>
<td>$308.00</td>
</tr>
<tr>
<td>118</td>
<td>2017-01-13</td>
<td>Conway</td>
<td>Coombs</td>
<td><a href="mailto:cccombsi@blogger.com">cccombsi@blogger.com</a></td>
<td>2810 Oakridge Park</td>
<td>32859</td>
<td>Florida</td>
<td>3529-1514-0357-9120</td>
<td>$60.00</td>
</tr>
<tr>
<td>119</td>
<td>2017-01-14</td>
<td>Germaine</td>
<td>Bere</td>
<td><a href="mailto:gbere@bravesites.com">gbere@bravesites.com</a></td>
<td>82802 Oakridge Park</td>
<td>20041</td>
<td>District of Columbia</td>
<td>670961-0240-4054-9000</td>
<td>$95.00</td>
</tr>
<tr>
<td>120</td>
<td>2017-01-15</td>
<td>Davide</td>
<td>Tolcharde</td>
<td><a href="mailto:dtolcharde@redcross.org">dtolcharde@redcross.org</a></td>
<td>89 Continental Avenue</td>
<td>79165</td>
<td>Texas</td>
<td>5018-7748-4325-9510</td>
<td>$137.00</td>
</tr>
<tr>
<td>121</td>
<td>2017-01-16</td>
<td>Nigel</td>
<td>Arttharg</td>
<td><a href="mailto:narthargl@zimodo.com">narthargl@zimodo.com</a></td>
<td>31 Mbridge Point</td>
<td>22301</td>
<td>Virginia</td>
<td>560225-6965-2870-0000</td>
<td>$496.00</td>
</tr>
<tr>
<td>122</td>
<td>2017-01-17</td>
<td>Rickard</td>
<td>Trenholm</td>
<td><a href="mailto:rtrenholm@csblocal.com">rtrenholm@csblocal.com</a></td>
<td>93 Hoepker Parkway</td>
<td>70593</td>
<td>Louisiana</td>
<td>3541-5241-5383-9970</td>
<td>$760.00</td>
</tr>
<tr>
<td>123</td>
<td>2017-01-18</td>
<td>Judith</td>
<td>Dwane</td>
<td><a href="mailto:jdwanen@vk.com">jdwanen@vk.com</a></td>
<td>7914 Eliot Lane</td>
<td>14276</td>
<td>New York</td>
<td>5456-4410-0914-3180</td>
<td>$474.00</td>
</tr>
<tr>
<td>124</td>
<td>2017-01-19</td>
<td>Susan</td>
<td>Ilden</td>
<td><a href="mailto:sildeno@aol.com">sildeno@aol.com</a></td>
<td>25204 Huxley Road</td>
<td>21684</td>
<td>Maryland</td>
<td>3574-8586-6367-9920</td>
<td>$83.00</td>
</tr>
<tr>
<td>125</td>
<td>2017-01-20</td>
<td>Abbey</td>
<td>Tribgle</td>
<td><a href="mailto:atribgle@google.com.au">atribgle@google.com.au</a></td>
<td>47 Debra Pass</td>
<td>74184</td>
<td>Oklahoma</td>
<td>3538-6047-6315-7710</td>
<td>$513.00</td>
</tr>
<tr>
<td>126</td>
<td>2017-01-21</td>
<td>Zsazsa</td>
<td>Dunster</td>
<td><a href="mailto:zdunsterg@nature.com">zdunsterg@nature.com</a></td>
<td>7 Gerald Alley</td>
<td>40576</td>
<td>Kentucky</td>
<td>3562-0325-7709-3490</td>
<td>$952.00</td>
</tr>
<tr>
<td>127</td>
<td>2017-02-01</td>
<td>Grannath</td>
<td>Friatt</td>
<td><a href="mailto:gfratttr@seattletimes.com">gfratttr@seattletimes.com</a></td>
<td>774 Prairieview Circle</td>
<td>29225</td>
<td>South Carolina</td>
<td>3571-1171-9476-8780</td>
<td>$942.00</td>
</tr>
<tr>
<td>128</td>
<td>2017-02-02</td>
<td>Ross</td>
<td>Gaudin</td>
<td><a href="mailto:rgaudins@samsung.com">rgaudins@samsung.com</a></td>
<td>3102 Loeprich Trail</td>
<td>68197</td>
<td>Nebraska</td>
<td>5108-7578-4665-7110</td>
<td>$572.00</td>
</tr>
<tr>
<td>129</td>
<td>2017-02-02</td>
<td>Aluino</td>
<td>Drovert</td>
<td><a href="mailto:adrovert@dagonadesign.com">adrovert@dagonadesign.com</a></td>
<td>2717 Northridge Avenue</td>
<td>72199</td>
<td>Arkansas</td>
<td>670999-3171-8848-0000</td>
<td>$318.00</td>
</tr>
<tr>
<td>130</td>
<td>2017-02-03</td>
<td>Shurlock</td>
<td>Braker</td>
<td><a href="mailto:sbrakeru@huffingtonpost.com">sbrakeru@huffingtonpost.com</a></td>
<td>30783 Jenna Alley</td>
<td>80945</td>
<td>Colorado</td>
<td>6331106-1894-9878-0000</td>
<td>$166.00</td>
</tr>
</tbody>
</table>
There is only one order on disk
\[ P(\text{Start Date} < X < \text{End Date} \& Y < \text{Start Amount}) \]
Models and Layout Can Be Data and Workload Dependent
Initial Results

Select-Project-Aggregate Queries over TPC-H line-item table. This are NOT TPC-H results!
Data size: 5GB
In-Memory Column Store, Single Threaded, Traditional Data Compression, No SIMD Opt.
Initial Results

Select-Project-Aggregate Queries over TPC-H line-item table. This are NOT TPC-H results!
Data size: 5GB
In-Memory Column Store, Single Threaded, Traditional Data Compression, No SIMD Opt.

Note: We are still at the beginning!
Sorting

(a) CDF Model Pre-Sorts
Sorting

(a) CDF Model Pre-Sorts

(b) Compact & local sort
Initial Results

Data: 64-bit doubles
Fundamental Algorithms & Data Structures

Our initial paper (CDF-based)
- Tree
- Hash-Map
- Bloom-Filter

Work in Progress (CDF-based)
- Multi-Dim Index
- Sorting
- Range-Filter
- DNA-Search
- CDF-Synth.

Work in Progress (Oracle/Full)
- Data Cubes
- Scheduling
- SQL Query Optimizer
- Cache Policy
- Join
- Nearest Neighbor

F(x)
Fundamental Algorithms & Data Structures

Our initial paper (CDF-based)
- Tree
- Hash-Map
- Bloom-Filter

Work in Progress (CDF-based)
- Multi-Dim Index
- Sorting
- Range-Filter
- DNA-Search
- CDF-Synth.

Work in Progress (Oracle/Full)
- Data Cubes
- Scheduling
- SQL Query Optimizer

- Cache Policy
- Join
- Nearest Neighbor

F(x)
How Would You Design Your Algorithms/Data Structure If You Have a Model for the Empirical Data Distribution?
We Need To Rewrite The Bible
We Need To Rewrite The Bibles
Data Science / Database Systems

• Index Structures
• Storage Manager
• Scheduling
• ...

Network Systems

• Router Table
• Rule Engine
• Bandwidth Optimization
• ...

Operating Systems

• Memory Management
• Scheduling
• Search
• ...

Mobile Systems

• Compression
• Location Services
• Transport Protocols
• ...
How We Develop Systems Has To Change Fundamentally
Customized Systems
Adaptation to data and workload
Big potential for GPUs/FPGAs/TPUs
New Lab on ML+System Co-Design at

DSAIL

Data Systems and AI Lab

Founding Sponsors
It Is the End of Systems As We Know Them

Tree  Hash-Map  Bloom-Filter  Multi-Dim Index  Sorting  Range-Filter  DNA-Search

Data Cubes  Scheduling  SQL Query Optimizer  Cache Policy  Join  Nearest Neighbor

Tim Kraska
<kraska@mit.edu>