The Case for Learned Index Structures

Authors: Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, Neoklis Polyzotis

Presenter: Ruijia Mao
Agenda

- Introduction
- Range Indexes - B-Tree Index
- Point Index - Hash-Map Index
- Existence Index - Bloom Filter Index
- Conclusion & Future Work
Introduction
Introduction: Index Examples

- B-Tree Index
- Hash-Map Index
- Bloom Filter Index
Introduction: B-Tree Index
Introduction:
Hash-Map Index

keys: John Smith, Lisa Smith, Sam Doe, Sandra Dee, Ted Baker
buckets:
- 000
- 001
- 002
- 151
- 152
- 153
- 154
- 253
- 254
- 255
entries:
- Lisa Smith 521-8976
- John Smith 521-1234
- Sandra Dee 521-9655
- Ted Baker 418-4165
- Sam Doe 521-5030
Introduction: Bloom Filter
Introduction:
Indexes are models

- General purpose index structures assume nothing about data distribution

- **Learned indexes** - learn a model that reflects patterns in the data - automatic synthesis of specialized index structures
Introduction:
Indexes are models

- Indexes are to a large extent learned models
- B-Tree Index - take a key as an input and predicts the position of a data record in a sorted set
- Bloom Filter - binary classifier
Range Indexes
Range Indexes

- Maps a key to a position
- For efficiency, indexing only the first key of every page
Range Indexes

- The B-Tree is a model, or in ML terminology, a regression tree.
- It maps a key to a position with a min- and max-error, with a guarantee that the key can be found in that region if it exists.
Range Indexes

Key

BTree

pos

pos - 0  pos + pagezise

pos - min_err  pos + max_err

Model (e.g., NN)
Range Index Models are CDF Models

• A model that predicts the position given a key inside a sorted array effectively approximates the cumulative distribution function (CDF).

• \( p = F(\text{Key}) \times N \)

• \( p \) is the position estimate

• \( F(\text{Key}) \) is \( P(X \leq \text{Key}) \)
A Frist, Naive Learned Index

- Data: 200M web-server log records
- Goal: building a secondary index over the times-tamps using Tensorflow
- Model: trained a two-layer fully-connected neural network with 32 neurons per layer using ReLU activation functions
A Frist, Naive Learned Index: Results

• Model: $\approx 1250$ predictions per second, $\approx 80,000$ nano-seconds (ns) to execute the model with Tensorflow, without the search time

• B-Tree: traversal over the same data $\approx 300$ns

• Binary search the entire data: $\approx 900$ns
A First, Naive Learned Index: Problems

- Tensorflow is designed for larger model
- Last mile: B-Trees are good in overfitting the data with a few operations, while the models are good at approximate the general shape of a CDF
- B-Trees are extremely cache- and operation-efficient
A Frist, Naive Learned Index

Figure 2: Indexes as CDFs
The RM-Index

- In order to solve challenges mentioned above, the authors developed
  - Learning Index Framework (LIF)
  - Recursive Model Indexes (RMI)
  - Standard-error-based search strategies
The RM-Index: LIF

• Learning Index Framework (LIF)

• An index synthesis system: given an index specification, LIF generates different index configurations, optimizes them, and tests them automatically.
The RM-Index: RMI

• Recursive Model Index (RMI)

• A hierarchy of models. 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.
The RM-Index: RMI

Figure 3: Staged models
The RM-Index: RMI Benefits

- It separates model size and complexity from execution cost.
- It leverages the fact that it is easy to learn the overall shape of the data distribution.
- It effectively divides the space into smaller sub-ranges, like a B-Tree, to make it easier to achieve the required “last mile” accuracy with fewer operations.
- There is no search process required in-between the stages.
The RM-Index: Hybrid Indexes

- Another advantage of the recursive model index is that mixtures of models can be built.
  - Top layer - a small ReLU neural net
  - Bottom - linear regression
The RM-Index:
Search Strategy

• **Model Biased Search** - the first middle point is set to the value predicted by the model

• **Biased Quaternary Search** - three middle points of quaternary search as $\text{pos} - \sigma$, $\text{pos}$, $\text{pos} + \sigma$
## Results

<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) Lookup (ns) Model (ns)</td>
<td>Size (MB) Lookup (ns) Model (ns)</td>
<td>Size (MB) Lookup (ns) Model (ns)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Btree</td>
<td>page size:</td>
<td>52.45 (4.00x) 274 (0.97x) 198 (72.3%)</td>
<td>51.93 (4.00x) 276 (0.94x) 201 (72.7%)</td>
<td>49.83 (4.00x) 274 (0.96x) 198 (72.1%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>26.23 (2.00x) 277 (0.96x) 172 (62.0%)</td>
<td>25.97 (2.00x) 274 (0.95x) 171 (62.4%)</td>
<td>24.92 (2.00x) 274 (0.96x) 169 (61.7%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>page size:</td>
<td>13.11 (1.00x) 265 (1.00x) 134 (50.8%)</td>
<td>12.98 (1.00x) 260 (1.00x) 132 (50.8%)</td>
<td>12.46 (1.00x) 263 (1.00x) 131 (50.0%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>6.56 (0.50x) 267 (0.99x) 114 (42.7%)</td>
<td>6.49 (0.50x) 266 (0.98x) 114 (42.9%)</td>
<td>6.23 (0.50x) 271 (0.97x) 117 (43.2%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>page size:</td>
<td>3.28 (0.25x) 286 (0.93x) 101 (35.3%)</td>
<td>3.25 (0.25x) 291 (0.89x) 100 (34.3%)</td>
<td>3.11 (0.25x) 293 (0.90x) 101 (34.5%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>101 (35.3%)</td>
<td></td>
<td>100 (34.3%)</td>
<td></td>
<td>101 (34.5%)</td>
<td></td>
</tr>
<tr>
<td>Learned Index</td>
<td>2nd stage models:</td>
<td>10k 0.15 (0.01x) 98 (2.70x) 31 (31.6%)</td>
<td>0.15 (0.01x) 222 (1.17x) 29 (13.1%)</td>
<td>0.15 (0.01x) 178 (1.47x) 26 (14.6%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50k 0.76 (0.06x) 85 (3.11x) 39 (45.9%)</td>
<td>0.76 (0.06x) 162 (1.60x) 36 (22.2%)</td>
<td>0.76 (0.06x) 162 (1.62x) 35 (21.6%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>100k 1.53 (0.12x) 82 (3.21x) 41 (50.2%)</td>
<td>1.53 (0.12x) 144 (1.81x) 39 (26.9%)</td>
<td>1.53 (0.12x) 152 (1.73x) 36 (23.7%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>200k 3.05 (0.23x) 86 (3.08x) 50 (58.1%)</td>
<td>3.05 (0.24x) 126 (2.07x) 41 (32.5%)</td>
<td>3.05 (0.24x) 146 (1.79x) 40 (27.6%)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Learned Index vs B-Tree
Point Index
Point Index:
Hash-map Index

- Conflict: too many distinct keys being mapped to the same position inside the Hash-map
Point Index: Hash-map Index

(a) Traditional Hash-Map

(b) Learned Hash-Map

Figure 7: Traditional Hash-map vs Learned Hash-map
Point Index: Hash-map Index

• Learning the CDF of the key distribution is one potential way to learn a better hash function.

• Use $h(K) = F(K) \ast M$, with key $K$ as our hash-function.

• If the model $F$ perfectly learned the empirical CDF of the keys, no conflicts would exist.
# Point Index: Results

<table>
<thead>
<tr>
<th></th>
<th>% Conflicts Hash Map</th>
<th>% Conflicts Model</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Data</td>
<td>35.3%</td>
<td>07.9%</td>
<td>77.5%</td>
</tr>
<tr>
<td>Web Data</td>
<td>35.3%</td>
<td>24.7%</td>
<td>30.0%</td>
</tr>
<tr>
<td>Log Normal</td>
<td>35.4%</td>
<td>25.9%</td>
<td>26.7%</td>
</tr>
</tbody>
</table>

*Figure 8: Reduction of Conflicts*
Existence Index
Existence Index: Learned Bloom Filters

- Separate keys from everything else
- Provide a specific FPR for realistic queries in particular while maintaining a FNR of zero
- Non-keys come from observable historical queries
- Use recurrent neural network (RNN)
Existence Index:
Learned Bloom Filters as a Classification Problem

(a) Traditional Bloom-Filter Insertion

(b) Learned Bloom-Filter Insertion

(c) Bloom filters as a classification problem

Key → Model

Model

Bloom filter

Yes → Database

No → Model

Yes
Figure 10: Learned Bloom filter improves memory footprint at a wide range of FPRs. (Here $W$ is the RNN width and $E$ is the embedding size for each character.)
Conclusion
Conclusion

• “In summary, we have demonstrated that machine learned models have the potential to provide significant benefits over state-of-the-art indexes, and we believe this is a fruitful direction for future research.”
Future Work

• Other ML Models
• Multi-dimensional Indexes
• Learned Algorithm - sorting or join
• GPU/TPU
Thanks

Q&A