Learned Indexes

Machine learning in databases: 07.03.18

Tyler Elliot Bettilyon

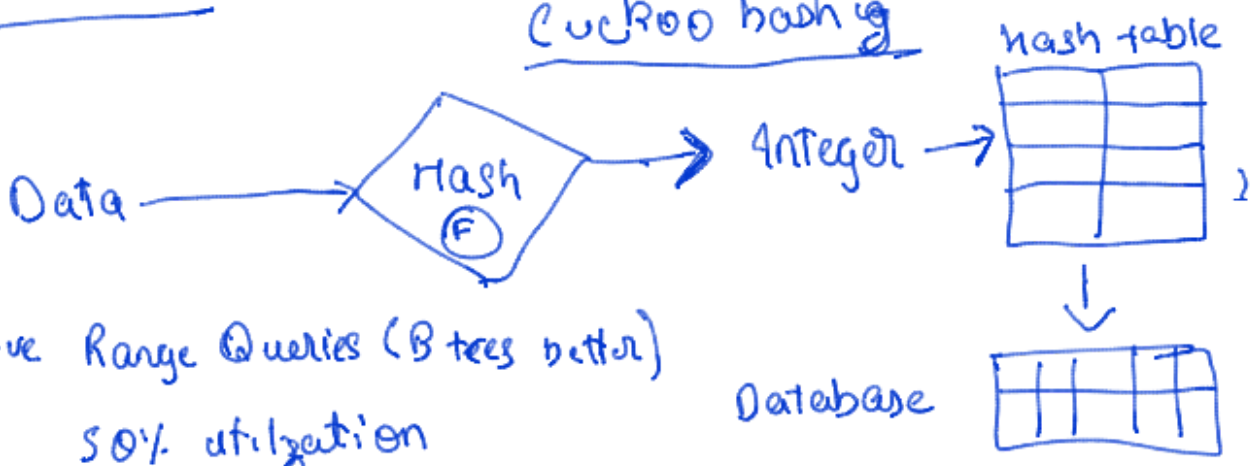
- HashTables
- BTrees

Performance of Index

1. Type of Query (Range - exact)
2. Lookup Speed
3. Cost to delete
4. Size & utilization
5. Minimize (IO ops) rather than Operations

Hash table

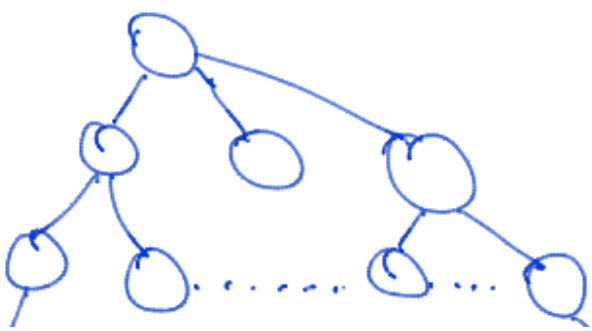
- Collision handling technique
  Cuckoo hashing



-ve Range Queries (B trees better)

50% utilization

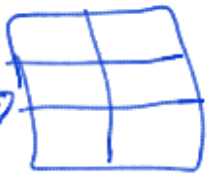
B-Tree



- Sorted
- Range Queries
- Query time (log n) / Search & Delete
- Must be balanced
- Split Operation (Insertion, Deletion) - Extra cost

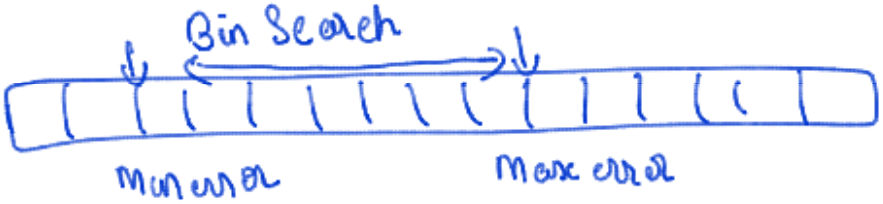
Learned Indexes

- Estimation ⇒ (func) * Key ⇒ Regression
- ML can be used?
- 

- Data → ML Model → int → 
hashtable

Adv.

- Takes into account distribution of data



- Small fully connected NN.

- 4n min

Btree vs Learned Indexes

- Speedup
- Better space improvement
- Save node size

Results & Conclusion

models take into account distribution

- Overfitting is the goal - Accuracy
- Data Warehousing application?
- Index using need not grow using ML model. Standard index grows.

- Moore law → x dead Classical CPU [x parallelization]
- ML models ⇒ parallelization
- Multidimension data ?

Q & A

- Suffix vs Btree
- Retrain on insertion of records (train in use)
- Normal distribution preferred?

  no → model trains for any distribution

- Could GAN can be used to generate indexes? (cluster)
- Parallelization in GPUs for databases is future?
- How ML model saves data?
  - → Hashmaps produces Uniform distribution
  - → ML takes into acc data distribution. produces fewer hash collision.
  - → Why GPU > CPU?
    - GPU has vector instructions
    - CPU does on 64 bit instruction
    - 512 bits < bit instruction, clock cycle is slower
    - Process 16 times faster, but clk is 4 times slower.

Last modified: 1:56 PM