PROBABILISTIC DATA STRUCTURES

INTRO

What is a Probabilistic Data Structure

TRADEOFFS

- Trade **accuracy** for **speed**

TRADEOFFS

- Trade **accuracy** for **speed**
    - meaning: they do not give 100% accurate results

TRADEOFFS

- Trade **accuracy** for **speed**
    - meaning: they do not give 100% accurate results
- Have less space requirements, also called sublinear

TRADEOFFS

- Trade **accuracy** for **speed**
  - meaning: they do not give 100% accurate results
- Have less space requirements, also called sublinear
  - meaning: to count N distinct items, the required space is less than N

TRADEOFFS

- Trade **accuracy** for **speed**
    - meaning: they do not give 100% accurate results
- Have less space requirements, also called sublinear
    - meaning: to count N distinct items, the required space is less than N
- Bonus: are associative

GOOD ENOUGH

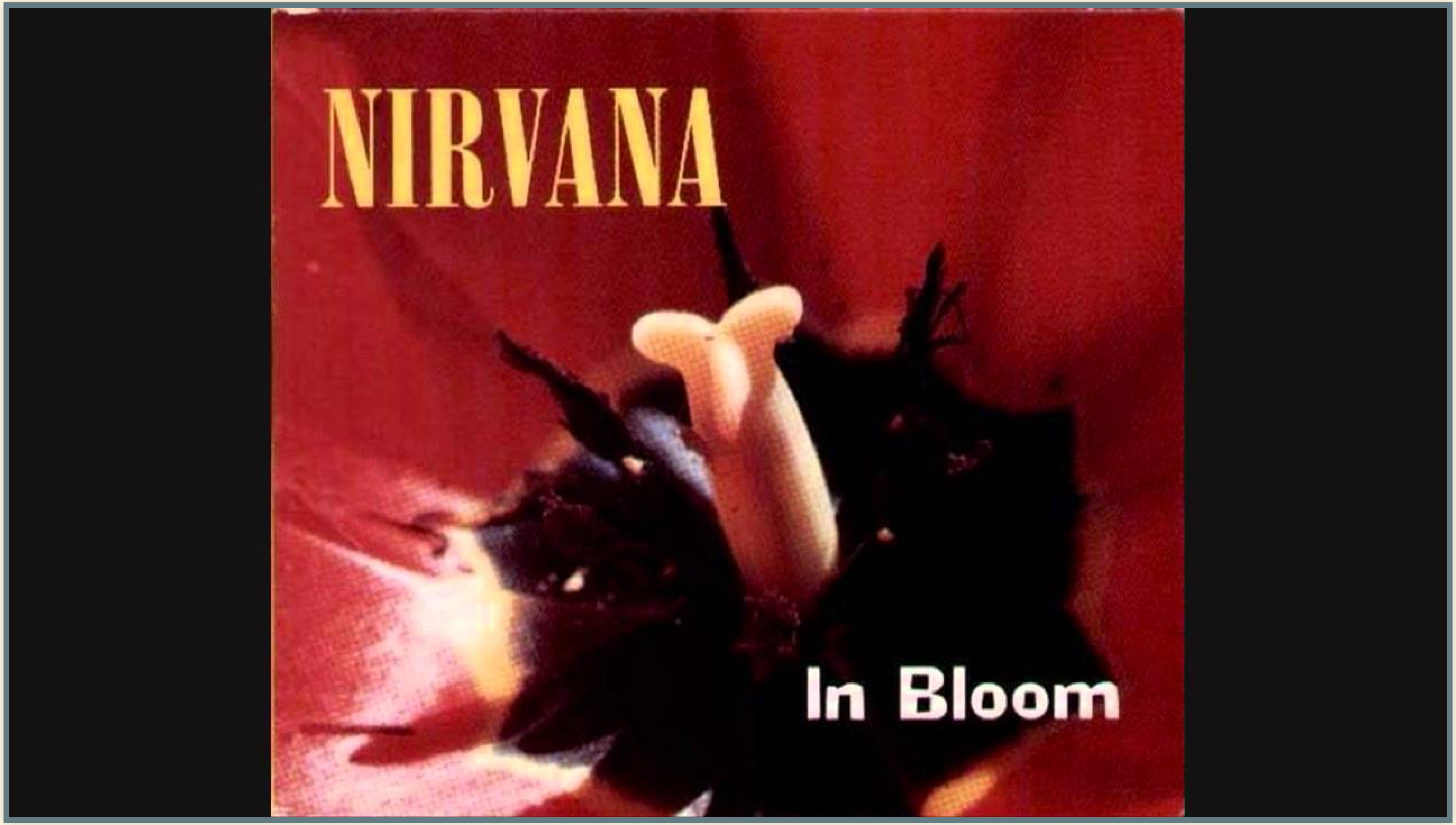
GOOD ENOUGH

- A query may return a wrong answer
    - But the answer is good enough (ex: count=1355, real count = 1299)

GOOD ENOUGH

- A query may return a wrong answer
  - But the answer is good enough (ex: count=1355, real count = 1299)
- Usually for BigData(tm) whatever that is

BLOOM FILTERS


NIRVANA
In Bloom

WHATS IT FOR?

Membership tests

Does this SET contains a particular ELEMENT?

An element is either MAYBE on the set, or IS NOT

FALSE POSITIVES ARE POSSIBLE

An element is either MAYBE on the set, or IS NOT

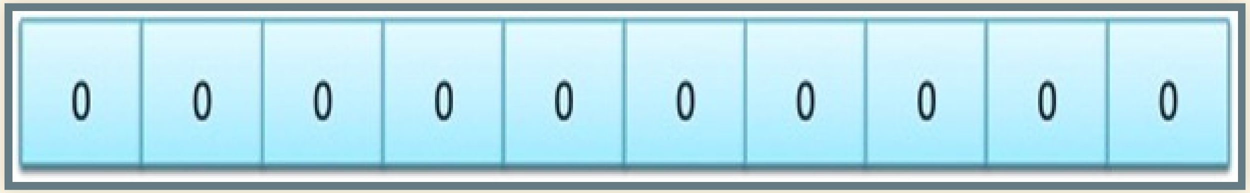
FALSE POSITIVES ARE POSSIBLE

FALSE NEGATIVES NEVER HAPPEN

An element is either MAYBE on the set, or IS NOT

HOW DOES IT WORK?

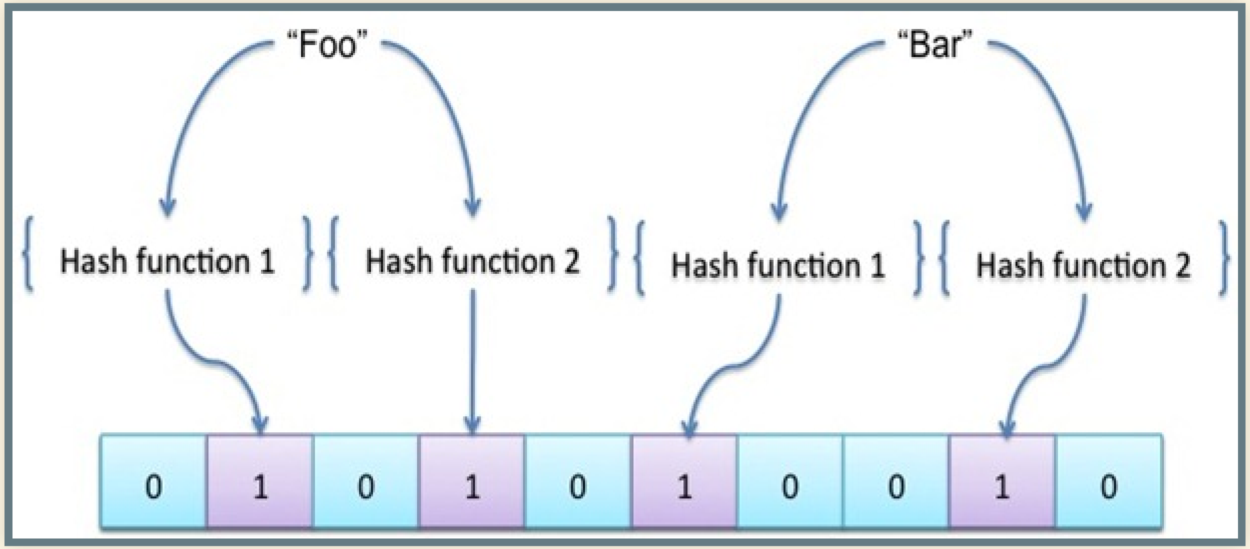
IT'S A BIT SET



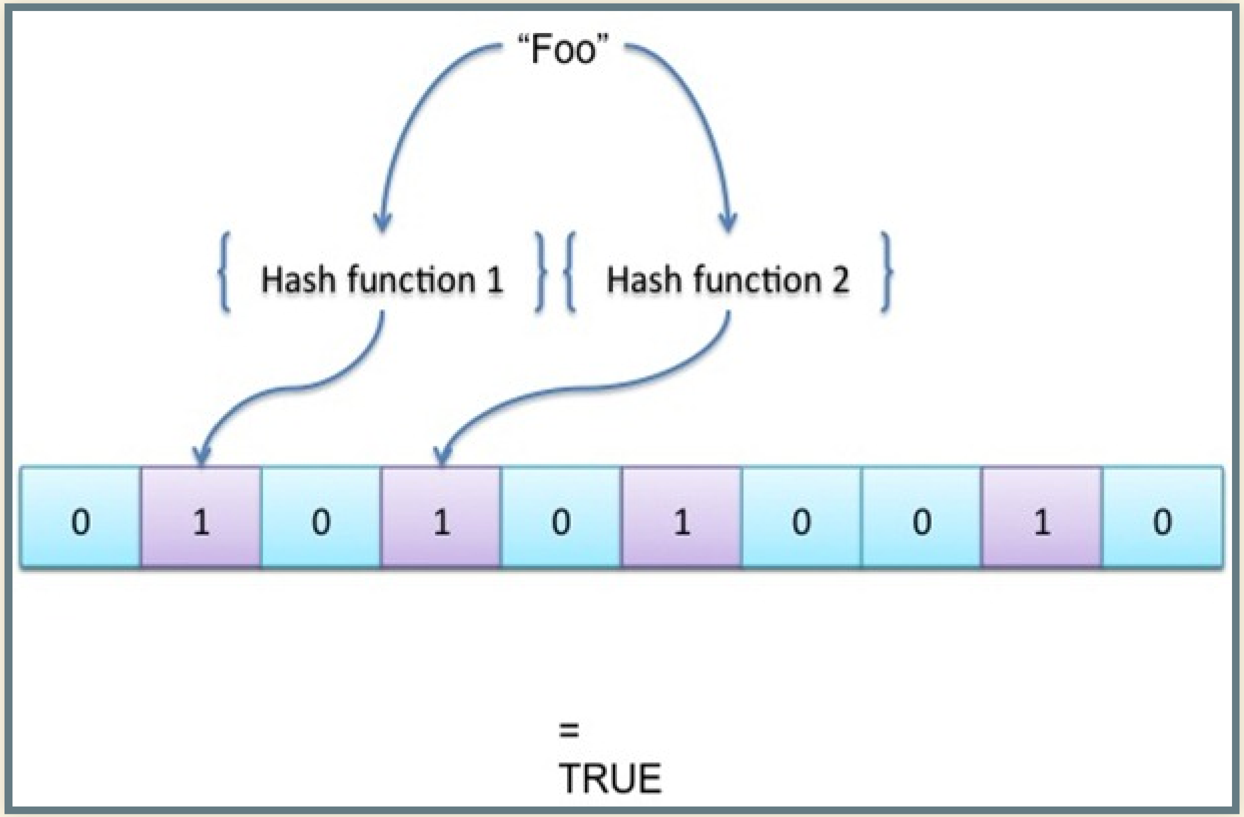
0	0	0	0	0	0	0	0	0	0

ADDING

Note that a very long string still occupies the same couple of bits.



QUERYING



"Foo"

Hash function 1

Hash function 2

0	1	0	1	0	1	0	0	1	0

=
TRUE

PARAMETERS

PARAMETERS

- Bitfield size (m)

PARAMETERS

- Bitfield size (m)
- Number of hash functions (k)
  - insertion and membership are O(k)

RULES

RULES

- One byte per item in the input set gives about a 2% false positive rate
  - 1024 elements to a 1KB Bloom Filter, about a 2% false positives

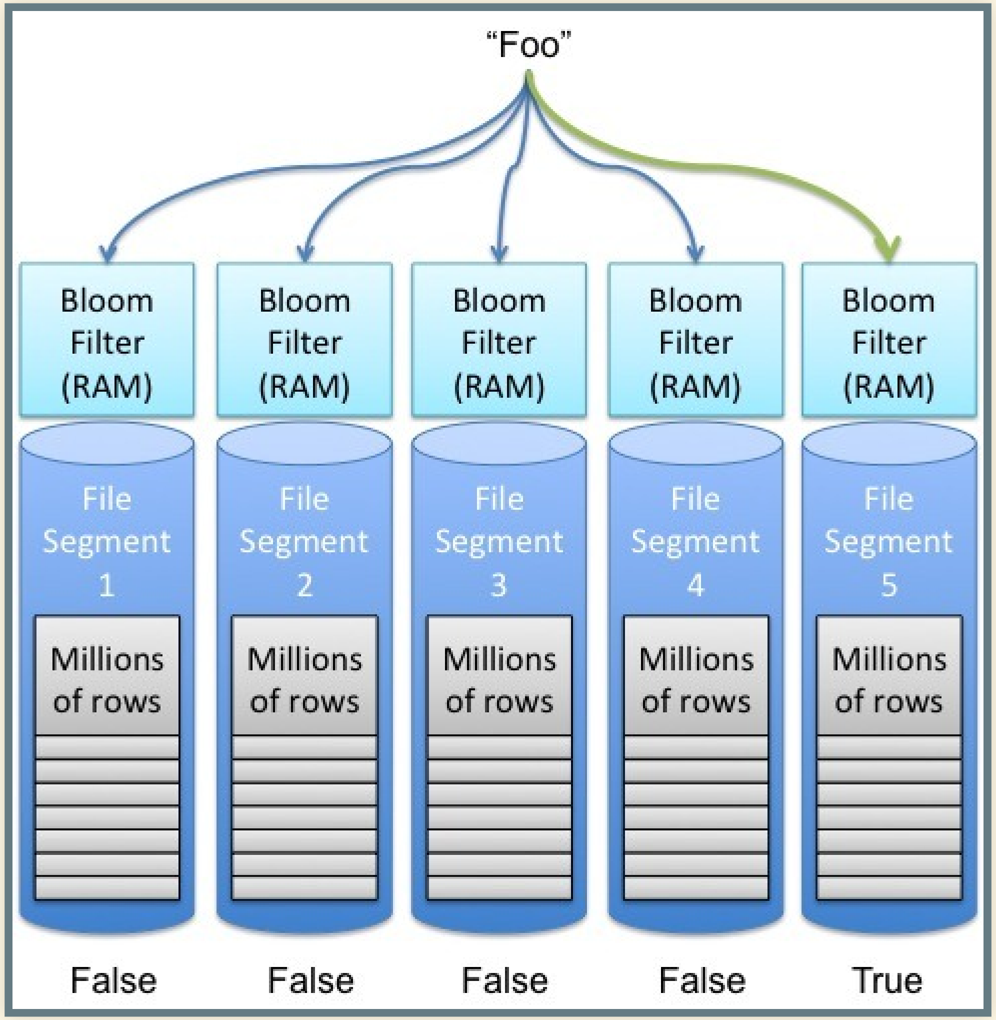
RULES

- One byte per item in the input set gives about a 2% false positive rate
    - 1024 elements to a 1KB Bloom Filter, about a 2% false positives
- The optimal number of hash functions is about 0.7 times the number of bits per item
    - 3 at a 10% false positive rate
    - 13 at a 0.01% false positive rate

RULES

- One byte per item in the input set gives about a 2% false positive rate
    - 1024 elements to a 1KB Bloom Filter, about a 2% false positives
- The optimal number of hash functions is about 0.7 times the number of bits per item
    - 3 at a 10% false positive rate
    - 13 at a 0.01% false positive rate
- The number of hashes dominates performance

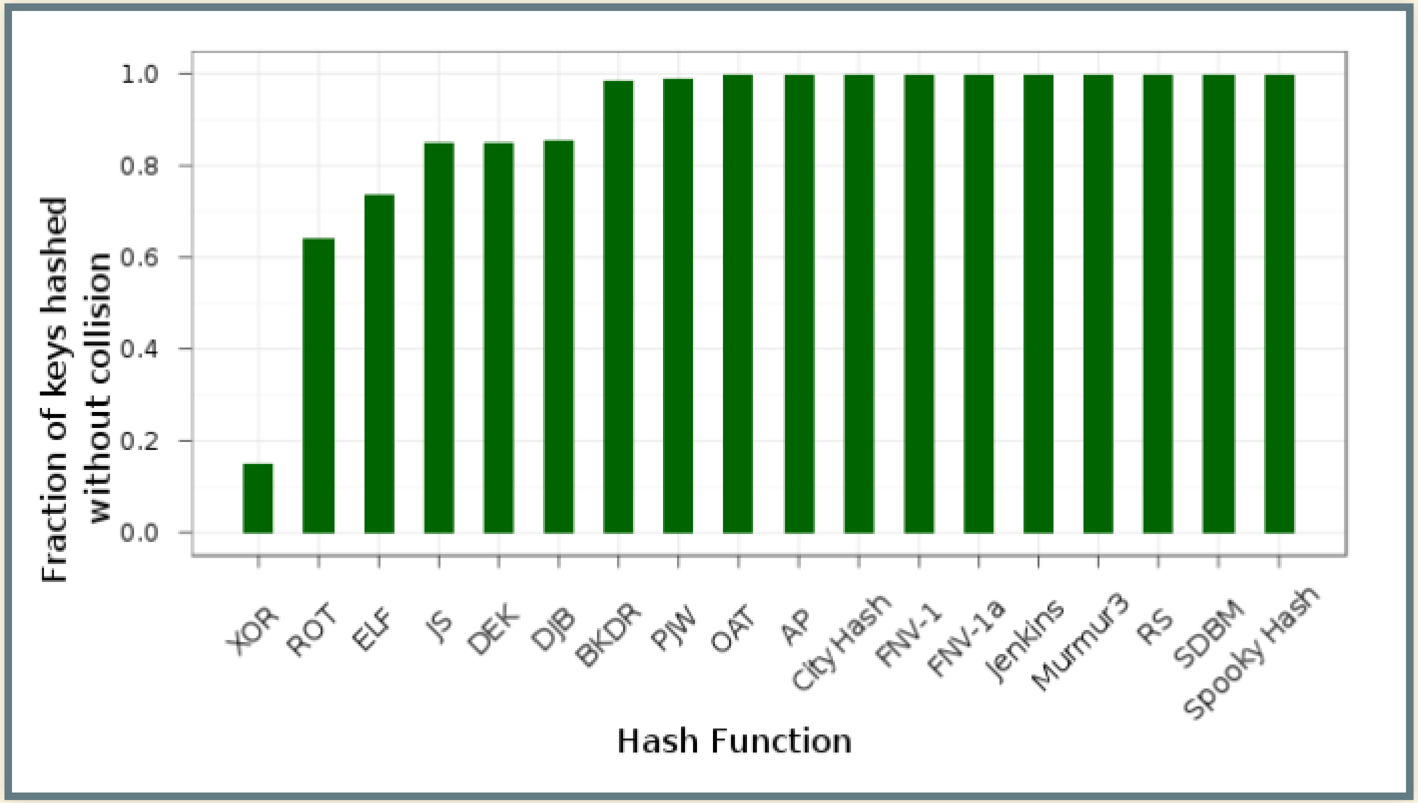
CASSANDRA


"Foo"
Bloom Filter (RAM)
Bloom Filter (RAM)
Bloom Filter (RAM)
Bloom Filter (RAM)
Bloom Filter (RAM)
File Segment 1
File Segment 2
File Segment 3
File Segment 4
File Segment 5
Millions of rows
Millions of rows
Millions of rows
Millions of rows
Millions of rows
False
False
False
False
True

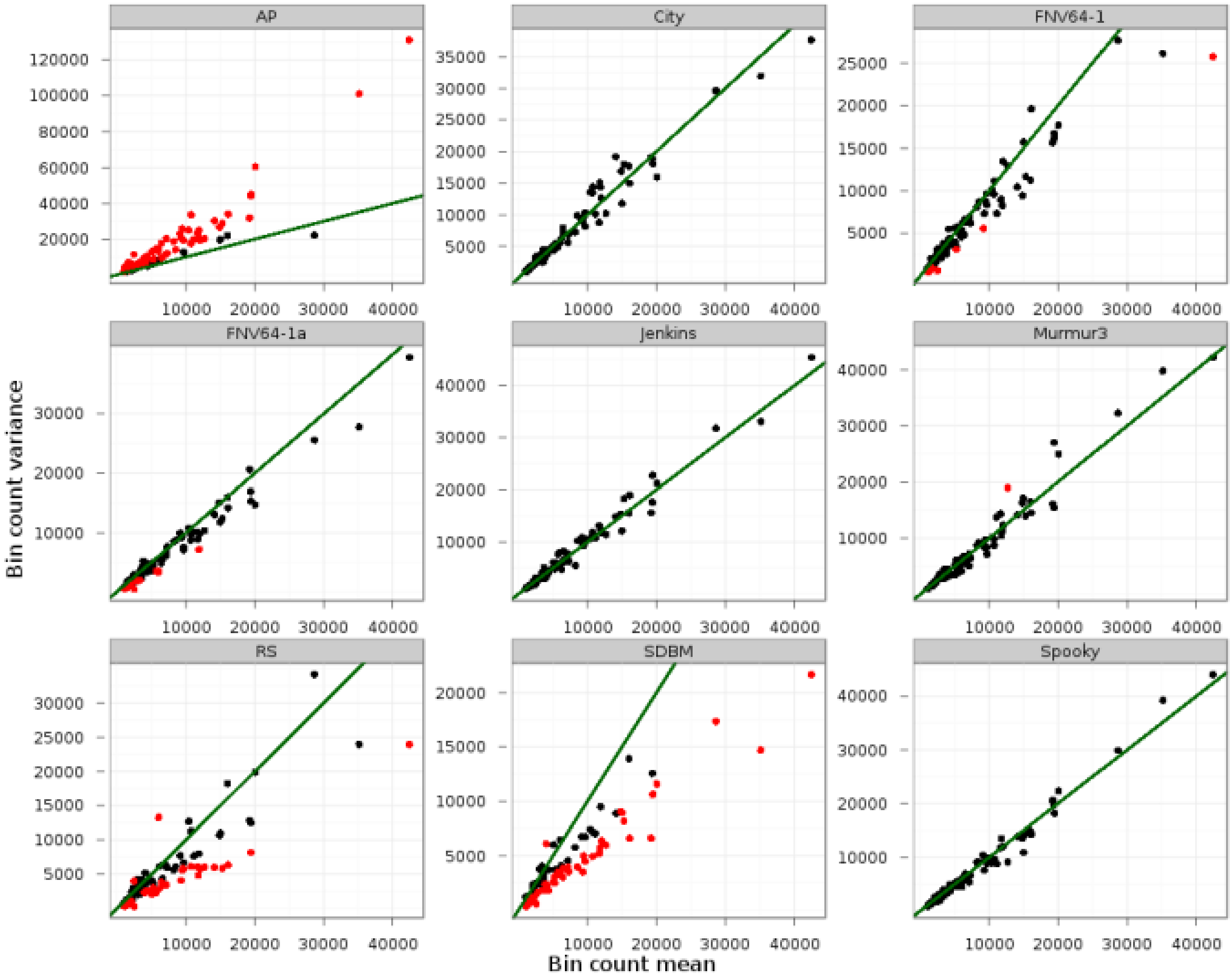
DIATRIBE: HASHING

JUST USE MURMUR3

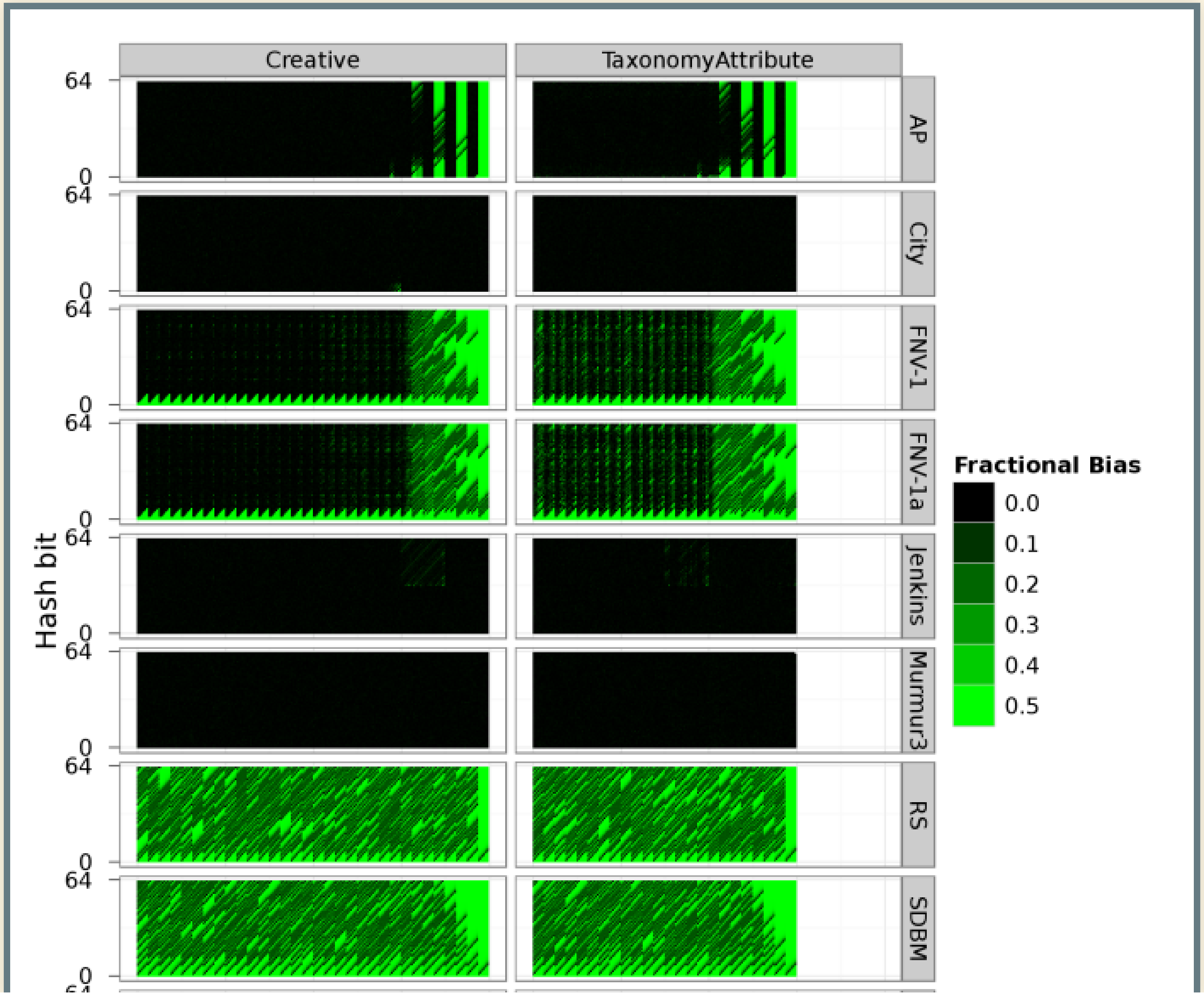
Fraction of keys hashed without collision (64 bits)

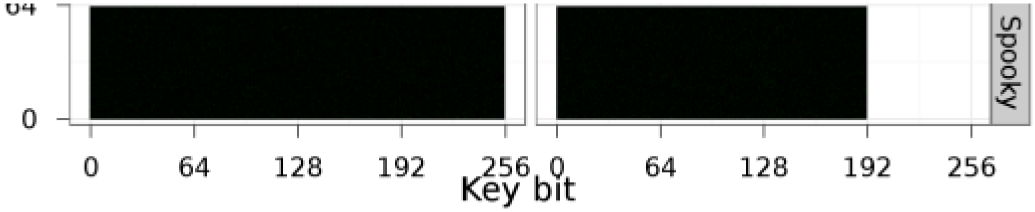


- V{n,i} = Number of items of namespace n hashed to the i-th bin
- Variance vs Mean for random distribution


AP
City
FNV64-1
FNV64-1a
Jenkins
Murmur3
RS
SDBM
Spooky
Bin count variance
Bin count mean
120000
100000
80000
60000
40000
20000
35000
30000
25000
20000
15000
10000
5000
10000
20000
30000
40000

BLACK=50% FLIP-PROBABILITY, BRIGHT GREEN=OUTPUT BIT IS “STUCK” - DOESN’T EVER VARY





64
0
0
64
128
192
256
0
64
128
192
256
Key bit
Spooky

COUNT MIN SKETCH

WHATS IT FOR?

Top-K frequencies/Heavy hitters

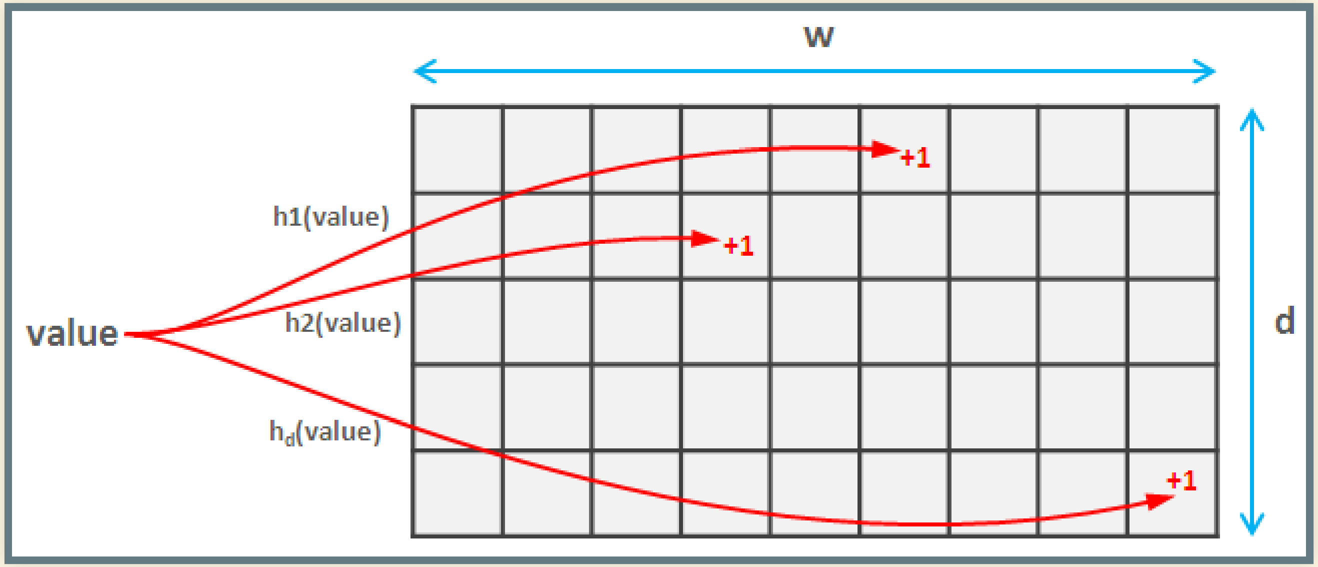
WHATS IT FOR?

Top-K frequencies/Heavy hitters

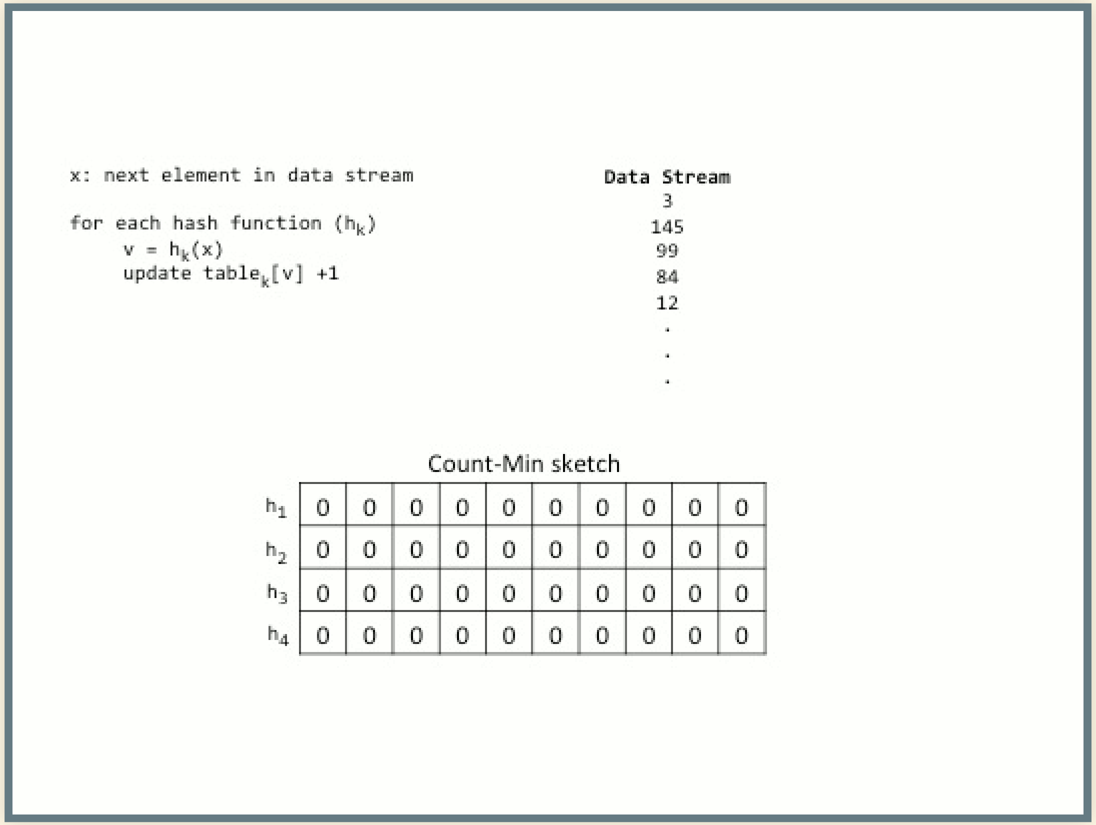
- How many times have you seen X?
    - Leaderboards
    - Stats
    - Rate limiting, packet stats, etc

HOW DOES IT WORK?

IT'S A 2D ARRAY



ADDING



```
x: next element in data stream

for each hash function ($h_k$)
    v = $h_k$(x)
    update $table_k$[v] +1
```

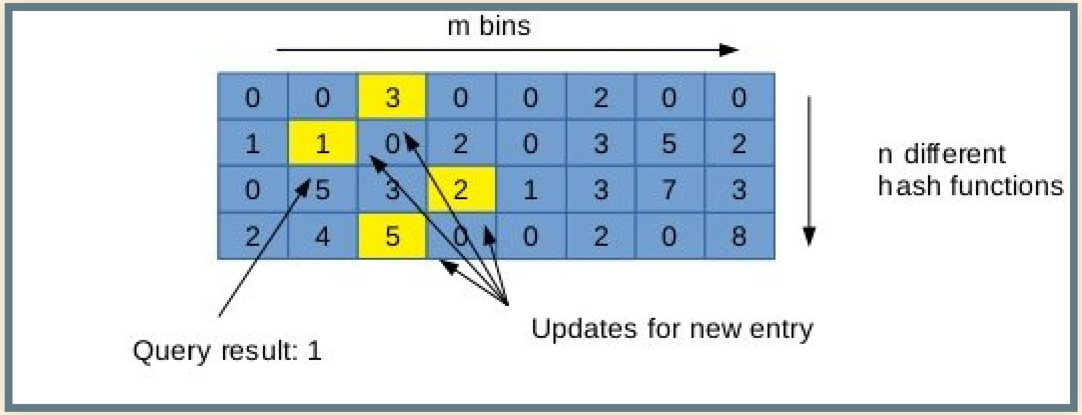
Data Stream

3
145
99
84
12
.
.
.

Count-Min sketch

h_1	0	0	0	0	0	0	0	0	0	0
h_2	0	0	0	0	0	0	0	0	0	0
h_3	0	0	0	0	0	0	0	0	0	0
h_4	0	0	0	0	0	0	0	0	0	0

QUERYING



0	0	3	0	0	2	0	0
1	1	0	2	0	3	5	2
0	5	3	2	1	3	7	3
2	4	5	0	0	2	0	8

- Take the minimum

PARAMETERS

- Number of hash functions
- Size of matrix

TDIGEST

WHATS IT FOR?

Quantiles

WHATS IT FOR?

Quantiles

- What's the 90% percentile for GET /my/service? and 99%?

WHATS IT FOR?

Quantiles

- What's the 90% percentile for GET /my/service? and 99%?
- anomaly detection: trigger at some percentile threshold

WHATS IT FOR?

Quantiles

- What's the 90% percentile for GET /my/service? and 99%?
- anomaly detection: trigger at some percentile threshold
- quantiles per metric per user/location/etc

WHATS IT FOR?

Quantiles

- What's the 90% percentile for GET /my/service? and 99%?
- anomaly detection: trigger at some percentile threshold
- quantiles per metric per user/location/etc
- Normally you need the full data set for a given quantile
    - You cannot calculate a quantile of quantiles - makes it hard to do streaming

HOW DOES IT WORK?

SPARSE REPRESENTATION OF THE CUMULATIVE DISTRIBUTION FUNCTION

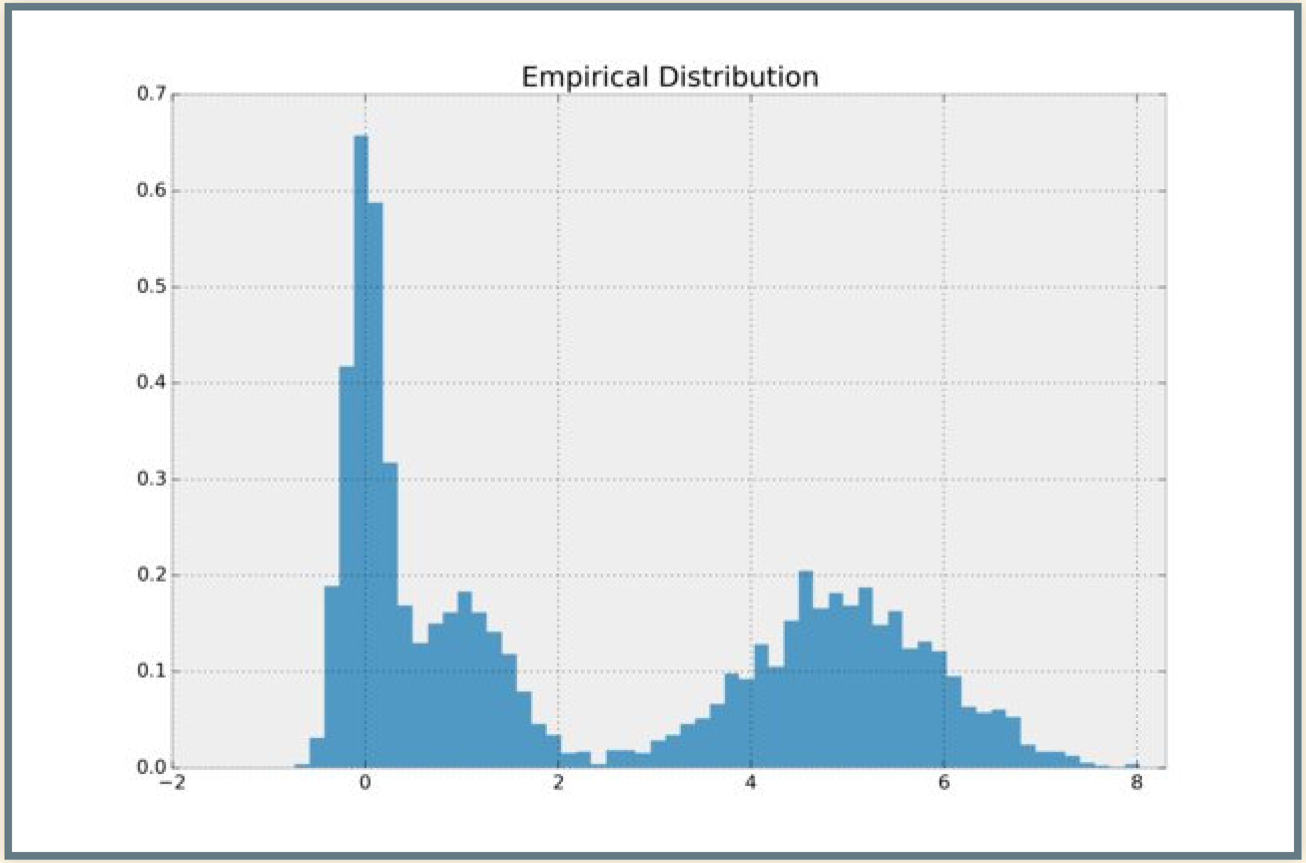
HOW DOES IT WORK?

SPARSE REPRESENTATION OF THE CUMULATIVE DISTRIBUTION FUNCTION

- After ingesting data, the data structure has learned the "interesting" points of the CDF, called centroids

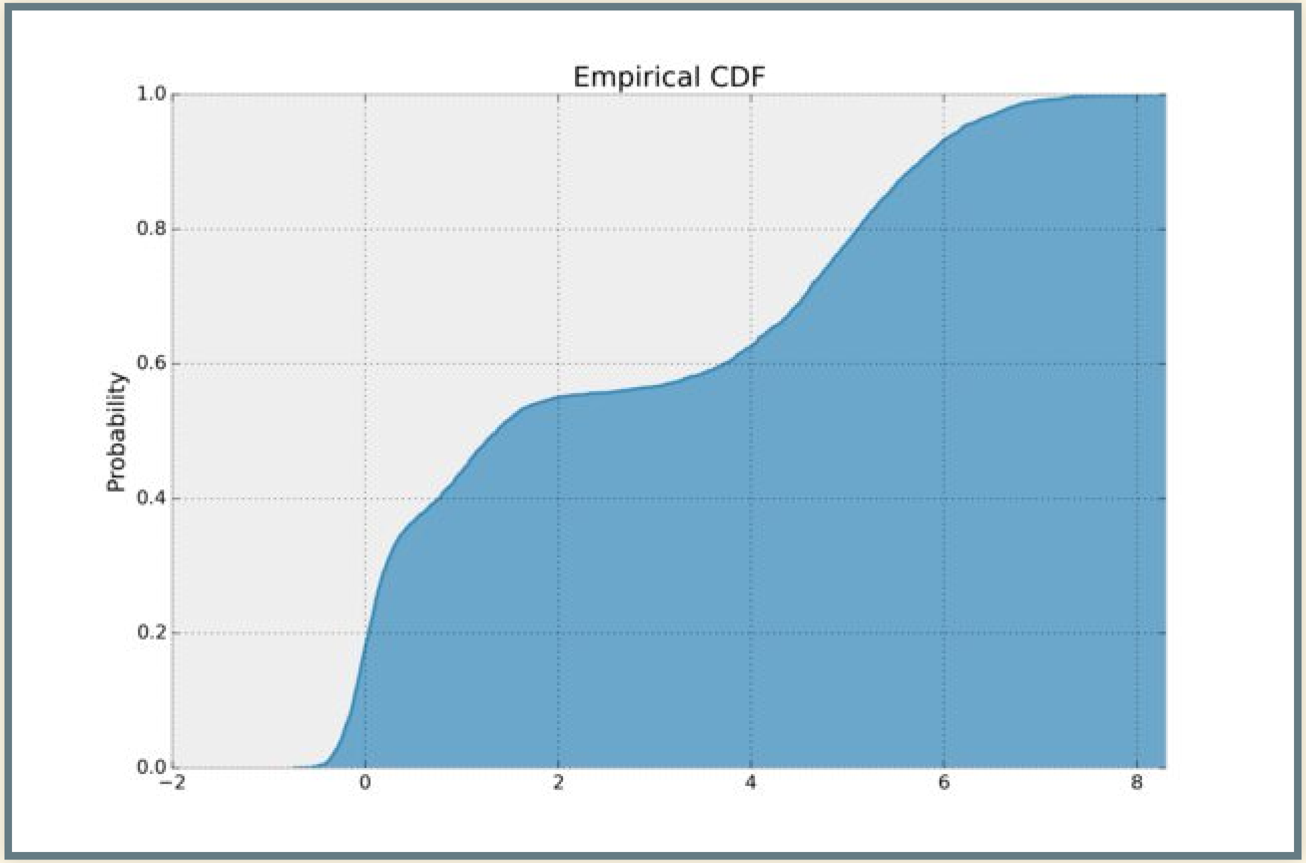
HOW DOES IT WORK?

SOME DATA



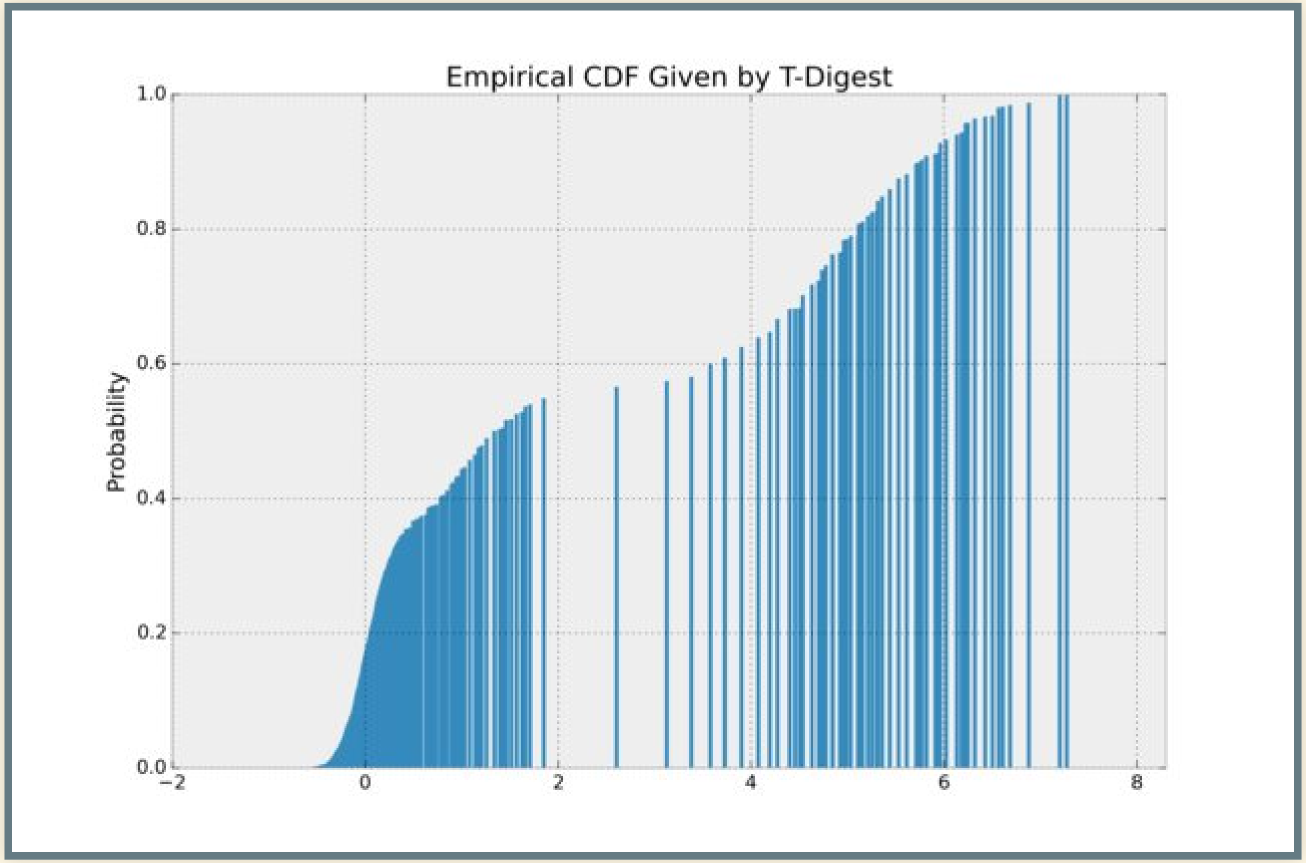
HOW DOES IT WORK?

EMPIRICAL CDF



HOW DOES IT WORK?

"INTERESTING" POINTS

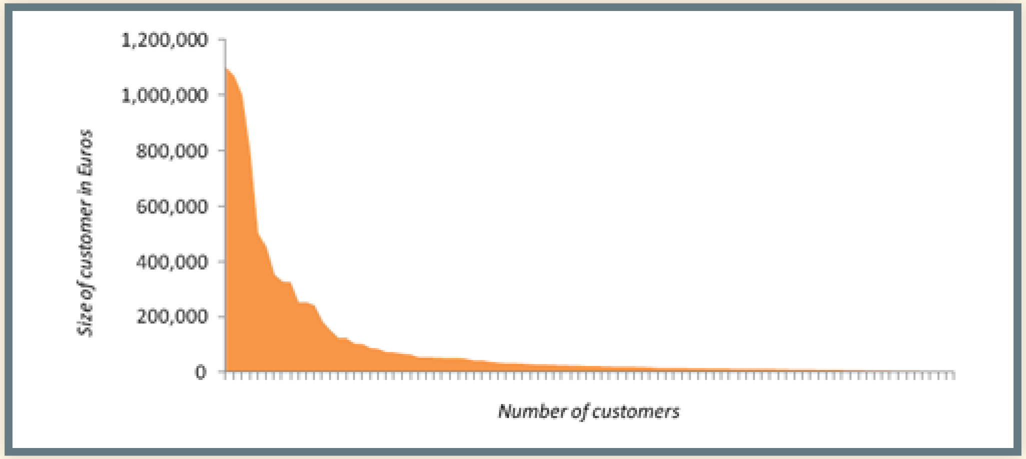


COMBINING

- Create a new t-Digest and treat the internal centroids of the two left-hand side digests as incoming data
- The resulting t-Digest is a only slightly larger, but more accurate

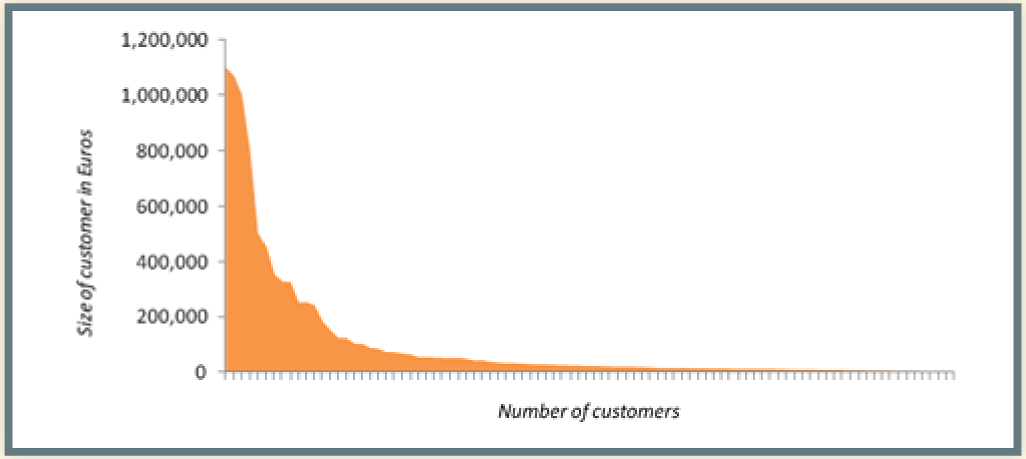
```
tDigest1 + tDigest2 =  tDigest3
-------------------    --------
incoming data       => new tDigest
```

QUERYING



- 8mb of pareto-distributed data into a t-Digest

QUERYING

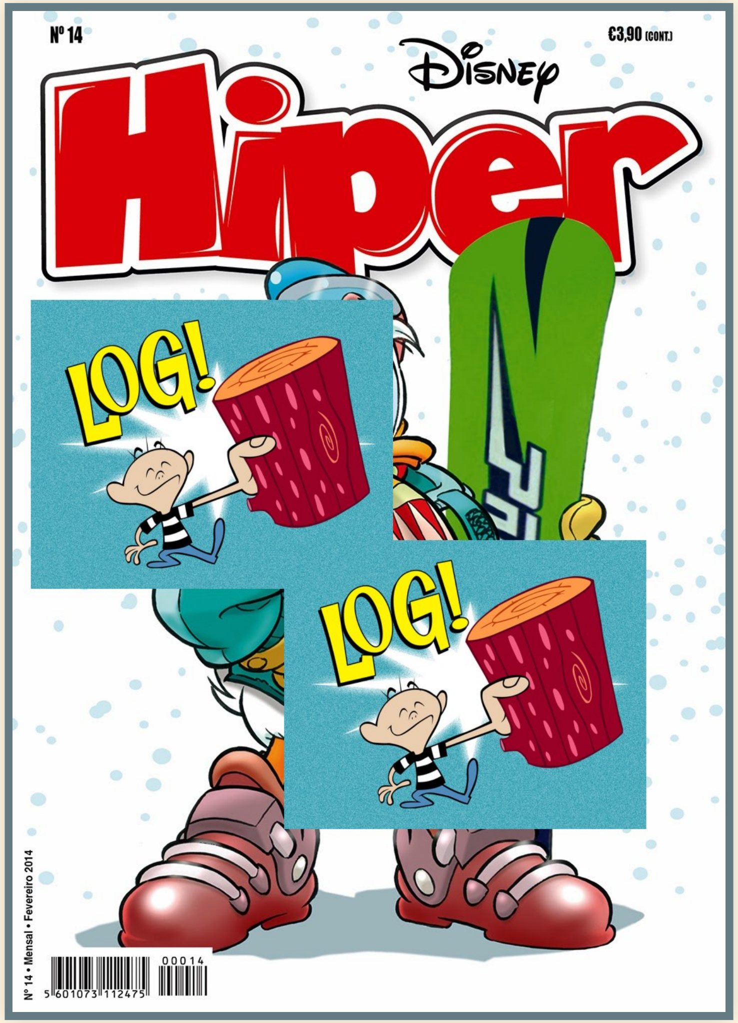


- 8mb of pareto-distributed data into a t-Digest
- Resulting size was 5kb
    - any percentile or quantile desired
    - accuracy was on the order of 0.002%.

PARAMETERS

- Compression
    - tradeoff of size vs accuracy
    - depends on the implementation, some expose more params than others
    - doesn't always mean the same thing

HYPERLOGLOG


Nº 14
€3,90 (CONT.)
Disney
Hiper
LOG!
LOG!
Nº 14 • Mensal • Fevereiro 2014
5 601073 112475 00014

WHATS IT FOR?

Cardinality Estimation

WHATS IT FOR?

Cardinality Estimation

- How many distinct ITEMS are there today? and yesterday? and the two days?
    - ex: unique visitors

WHATS IT FOR?

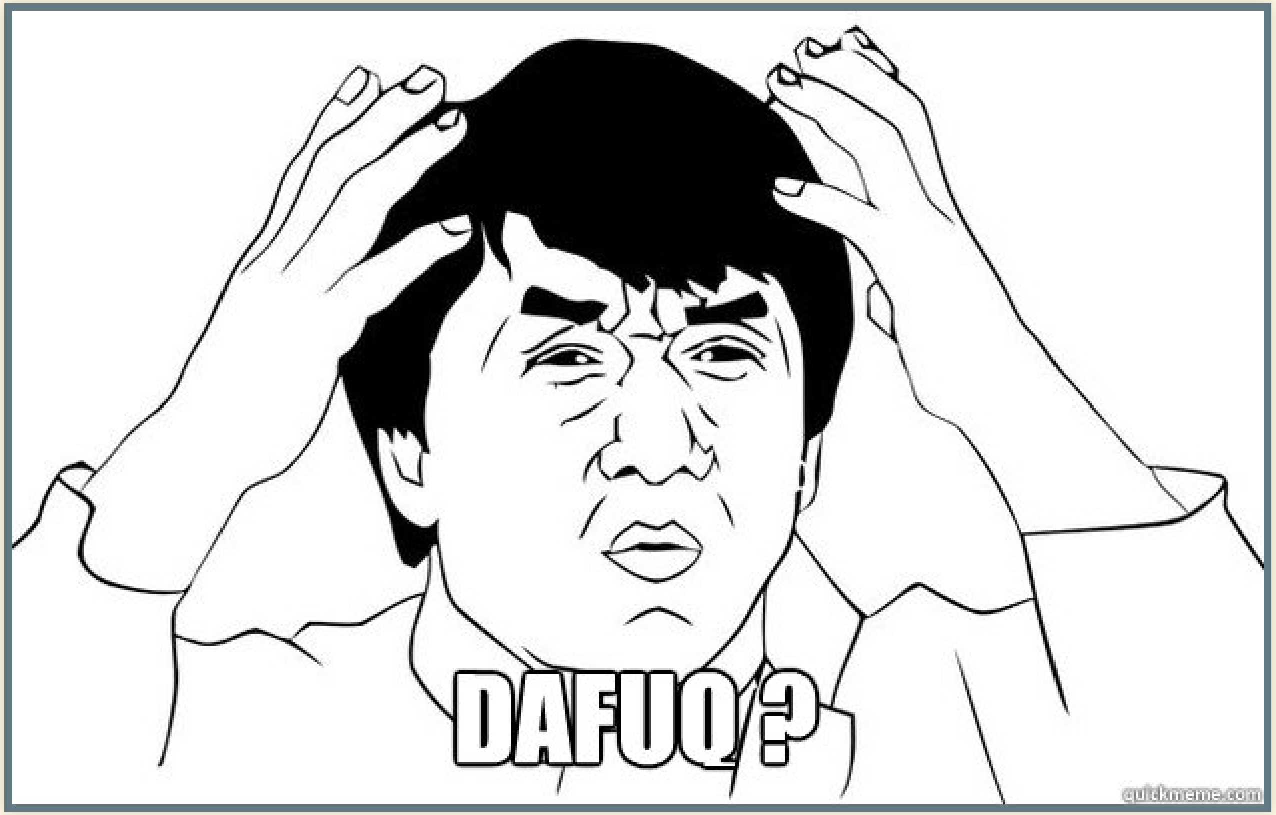
Cardinality Estimation

- How many distinct ITEMS are there today? and yesterday? and the two days?
    - ex: unique visitors
- group-by/count without keeping all the data

HOW DOES IT WORK?

IT'S COMPLICATED

> *...The observation that the cardinality of a multiset of uniformly-distributed random numbers can be estimated by calculating the maximum number of leading zeros in the binary representation of each number in the set.*
>
> *If the maximum number of leading zeros observed is n, an estimate for the number of distinct elements in the set is 2^n.*


DAFUQ ?
quickmeme.com

REAL SLOW NOW

- In a random stream of integers

REAL SLOW NOW

- In a random stream of integers
    - ~50% of the numbers (in binary) starts with "1"

REAL SLOW NOW

- In a random stream of integers
  - ~50% of the numbers (in binary) starts with "1"
  - 25% starts with "01"

REAL SLOW NOW

- In a random stream of integers
  - ~50% of the numbers (in binary) starts with "1"
  - 25% starts with "01"
  - 12,5% starts with "001"

REAL SLOW NOW

- In a random stream of integers
  - ~50% of the numbers (in binary) starts with "1"
  - 25% starts with "01"
  - 12,5% starts with "001"

If you observe a random stream and see a "001", there is a higher chance that this stream has a cardinality of 8.

BUCKETING

- number: 13,200,393
- hash: 2,005,620,294
- bits: [100010110101011001000110]

```
100010110101011001 000110
   ---------------|------
   value            index
```

BUCKETING

- number: 13,200,393
- hash: 2,005,620,294
- bits: [100010110101011001000110]

```
100010110101011001 000110
   ---------------|------
   value            index
```

- value: number of zeros +1 (rtl)

BUCKETING

- number: 13,200,393
- hash: 2,005,620,294
- bits: [100010110101011001000110]

```
100010110101011001 000110
   ---------------|------
   value            index
```

- value: number of zeros +1 (rtl)
- index: The lowest b bits used to determine the index of the register whose value is to be updated. (m=2b)

BUCKETING

- number: 13,200,393
- hash: 2,005,620,294
- bits: [10001011010101011001000110]

```
10001011010101011001 000110
   ----------------|------
   value             index
```

- value: number of zeros +1 (rtl)
- index: The lowest b bits used to determine the index of the register whose value is to be updated. (m=2b)
  - each bucket will serve as an "estimator"

ESTIMATING

- LogLog:
    - In order to compute the number of distinct values in the stream you would just take the average of all of the m buckets

    -
      ```
      distinct vals = constant * m * 2^(avg R)
      ```

ESTIMATING

- LogLog:
    - In order to compute the number of distinct values in the stream you would just take the average of all of the m buckets

    -
      ```
      distinct vals = constant * m * 2^(avg R)
      ```
- HyperLogLog uses
    - large range correction (??)
    - *Harmonic Mean* which tends to behave better for extreme values

HARMONIC WUT?

EXAMPLE

http://content.research.neustar.biz/blog/hll.html

UNIONS/INTERSECTIONS

How many distinct visitors we had in Monday AND Tuesday?

- Are lossless (for same HLL size)
  - Some guys tried to combine different HLL with different sizes

PARAMETERS

- number of buckets/registers
    - theoretical HLL error bounds (1.04 / sqrt(m))

THIS IS HUGE

Who's using?

- Node, Java, C, etc etc
- Postgres
- Redis
- Twitter Algebird, Scalding
- Druid (MPP)
- Basically anyone who needs to count distinct/group-by

CONCLUSION

CONCLUSION

- evaluate the scenario to see if approximation is useful

CONCLUSION

- evaluate the scenario to see if approximation is useful
- test the npm packages, some are just crap

CONCLUSION

- evaluate the scenario to see if approximation is useful
- test the npm packages, some are just crap
    - I would write them in C and use ffi

END