

Hash



A probabilistic approach for big data

Who am I?



Luca Mastrostefano
luca@translated.net

- Product manager of MyMemory at Translated
- IT background
- Algorithms lover

Syllabus

Problem	Use case
Fast and exact search	Databases - Search
Stream filter	Translated - MyMemory
Counting unique items in a stream	ClickMeter - IPs analysis
Probabilistic search	Memopal - Search for similar files

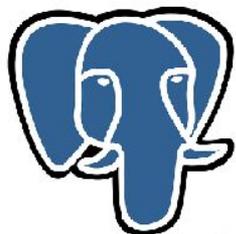
Search algorithms

...

Databases - Fast and exact search

Static, extendible and linear hash indexes

Use case

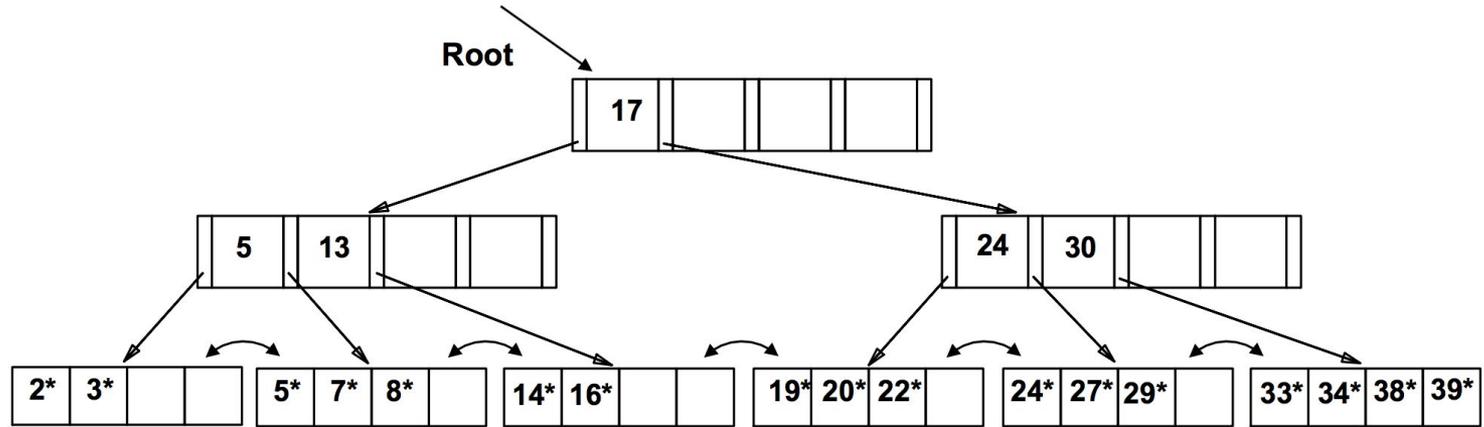


PostgreSQL



Sometimes also a logarithmic complexity is too expensive.

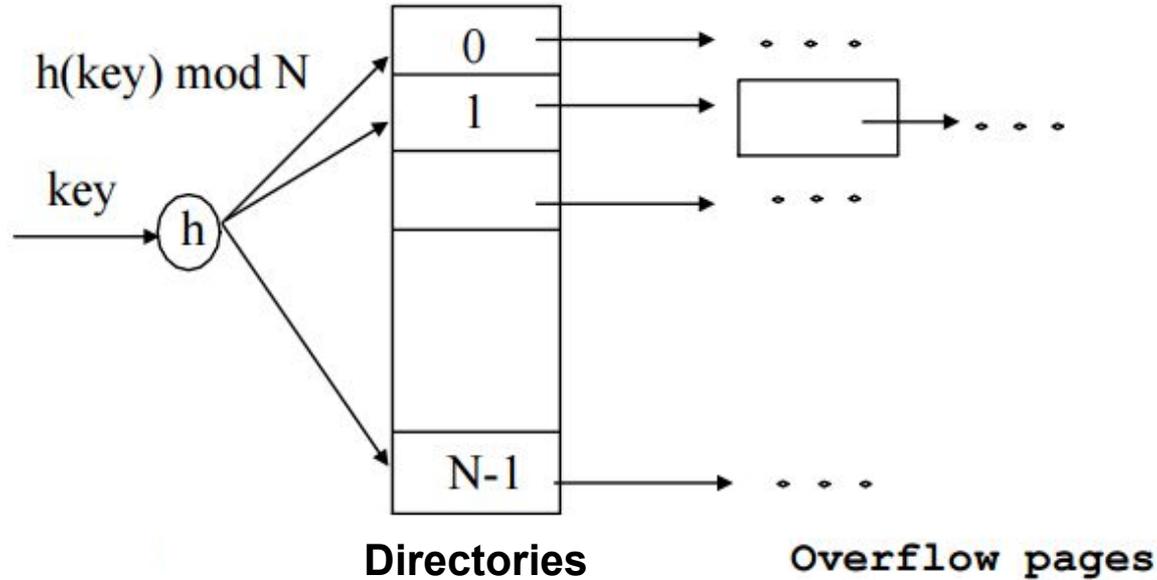
B⁺ tree index



Select/Insert $\cong \log_F(\# \text{ items})$

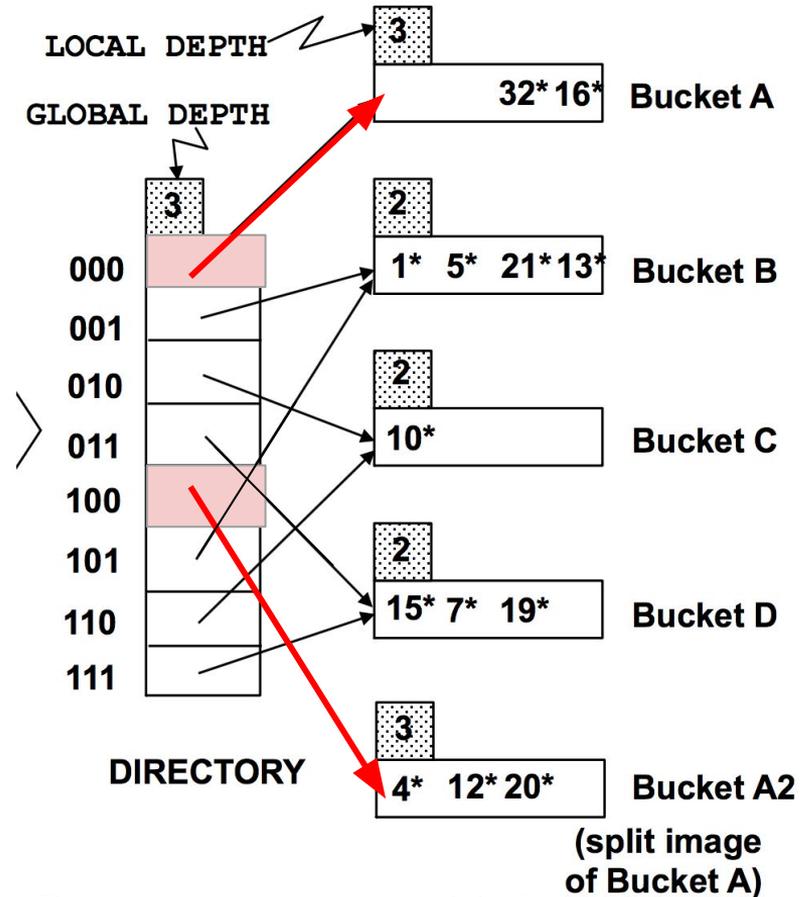
Search - Hash index

Static hash index



$$\text{Select/Insert} \cong 2 + (\# \text{ overflow pages})$$

Dynamic hash index - Extendible



$$\text{Select/Insert} \cong 2 + (\# \text{ overflow pages})$$

overflow pages almost constant

Dynamic hash index - Linear

Intuition:

- Avoid the directories to save one memory access.
- Split one bucket per time: **it fits real-time environments!**

$$\text{Select/Insert} \cong 1 + (\# \text{ overflow pages})$$

overflow pages almost constant

Indexes comparison - Secondary memory accesses

B⁺ tree index

Select/Insert $\approx \text{Log}$

4 accesses ≈ 30 ms

VS

Linear hash index

Select/Insert $\approx \text{const}$

1 access ≈ 7 ms

4x in case of billions of entries

Stream filter: $x \in U$?

...

Translated - MyMemory

Bloom filter

Use case

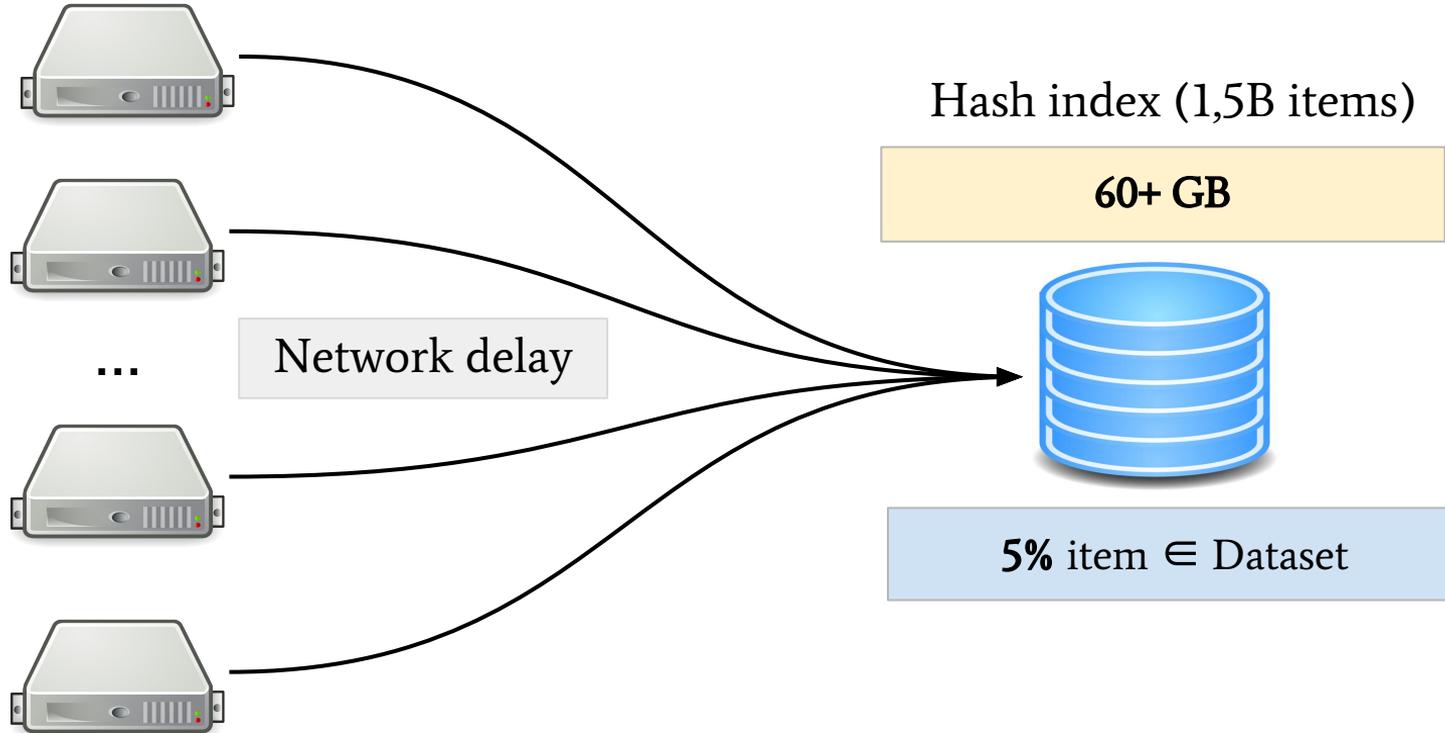


Ottieni una traduzione migliore grazie a **1.300.383.638** contributi umani

Italiano Inglese Più contesto

The delay introduced by the secondary memory does not fit an environment in which milliseconds matter.

Stream filter - Naïve approach

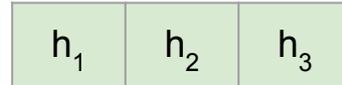


Stream filter - Bloom filter

Bloom filter - Insert



Bit array of length m

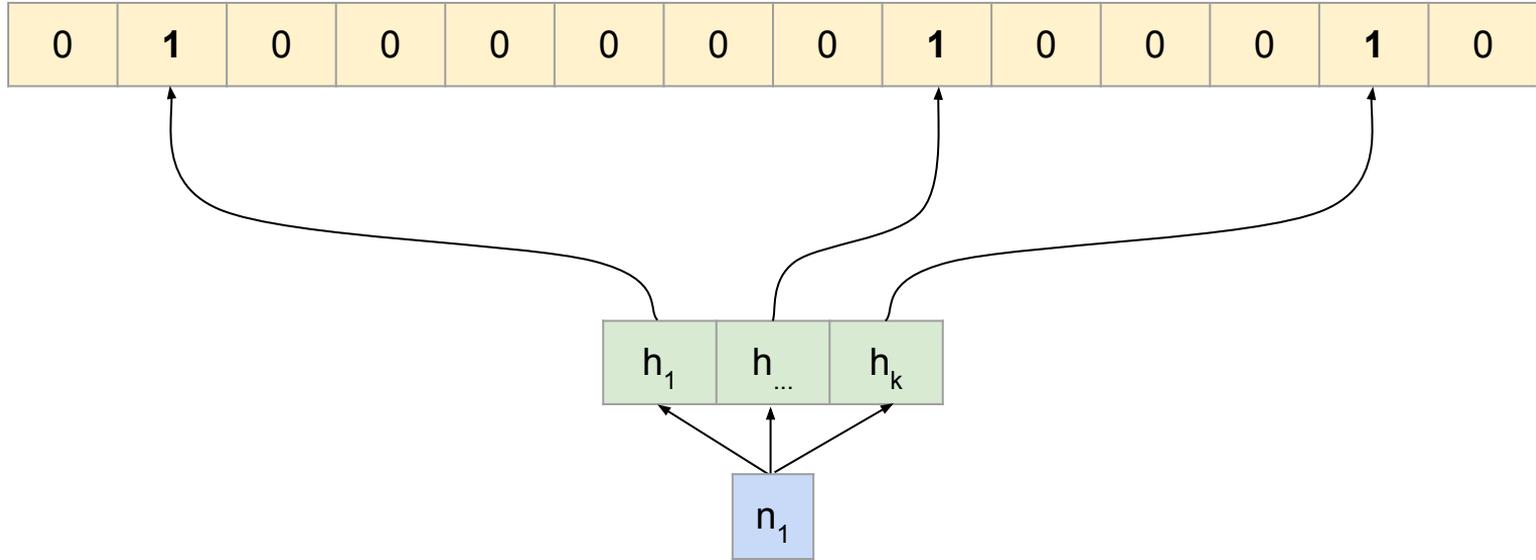


k hash functions

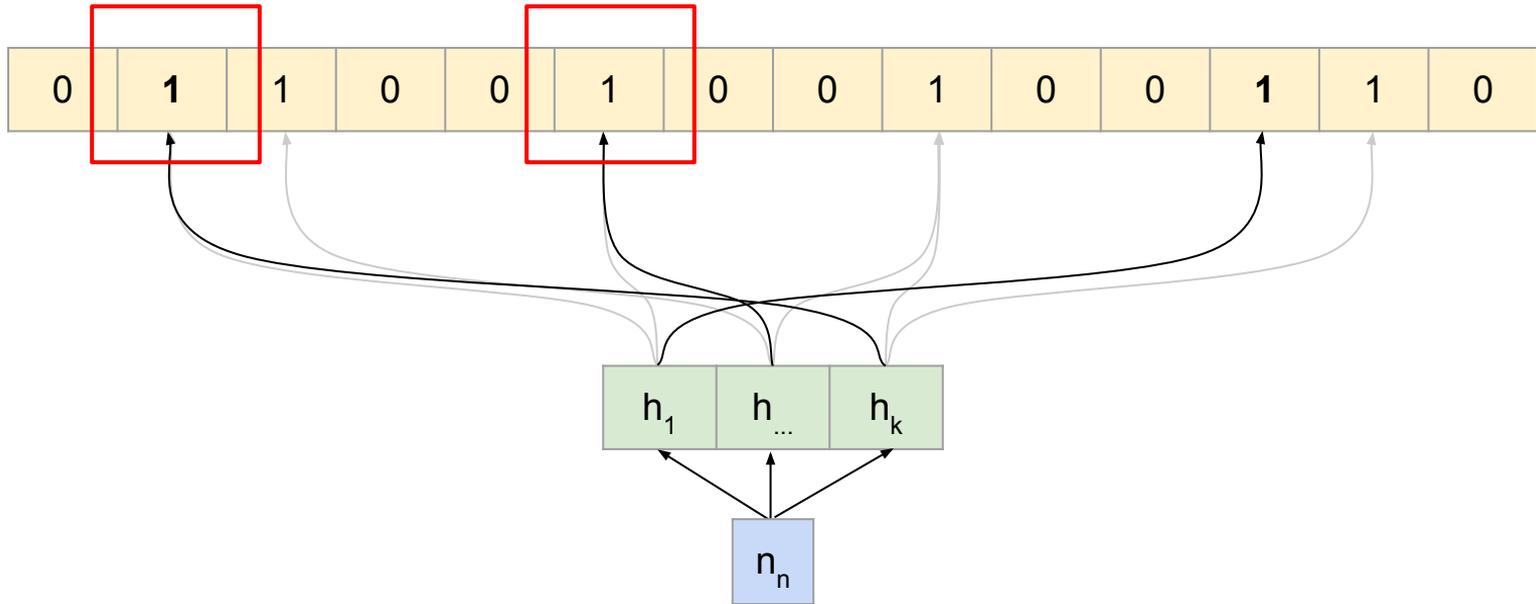


n items to insert

Bloom filter - Insert



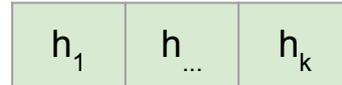
Bloom filter - Insert



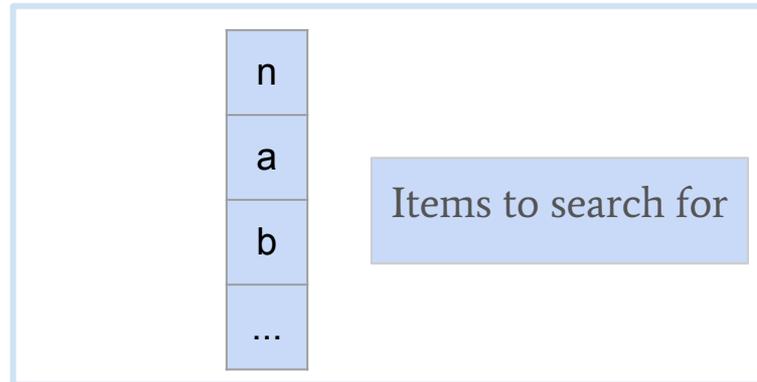
Bloom filter - Search



Fixed bit array

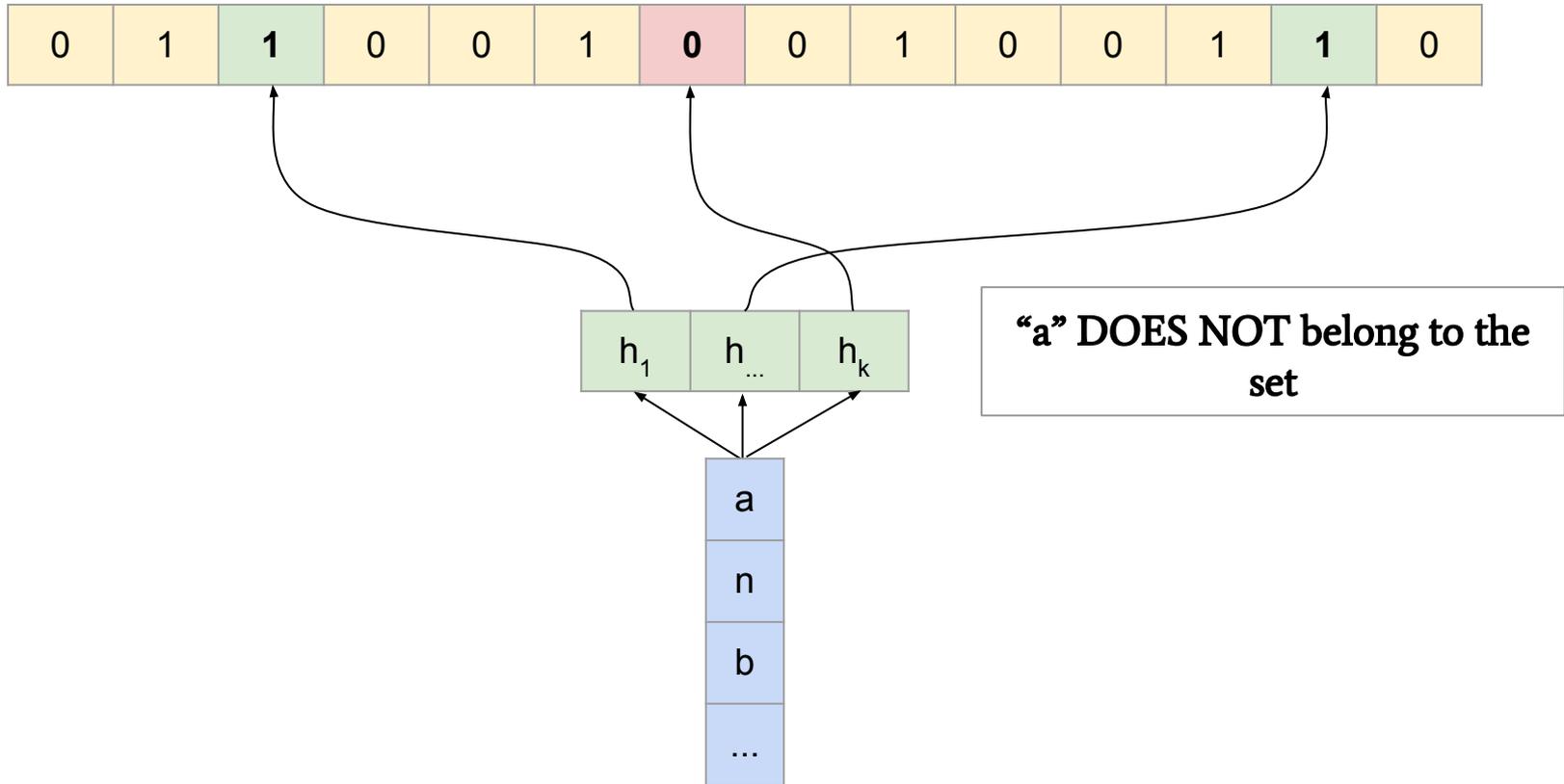


Same hash functions

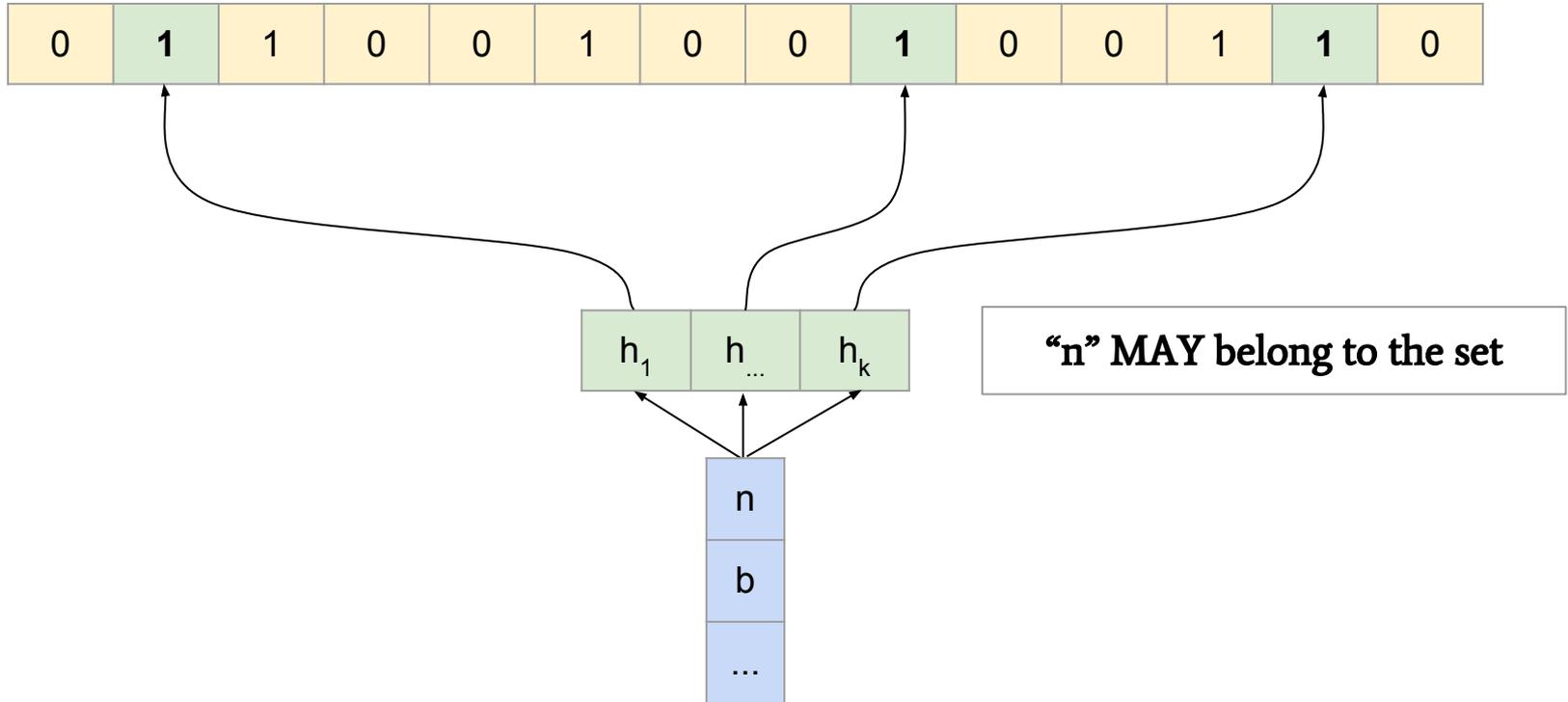


Items to search for

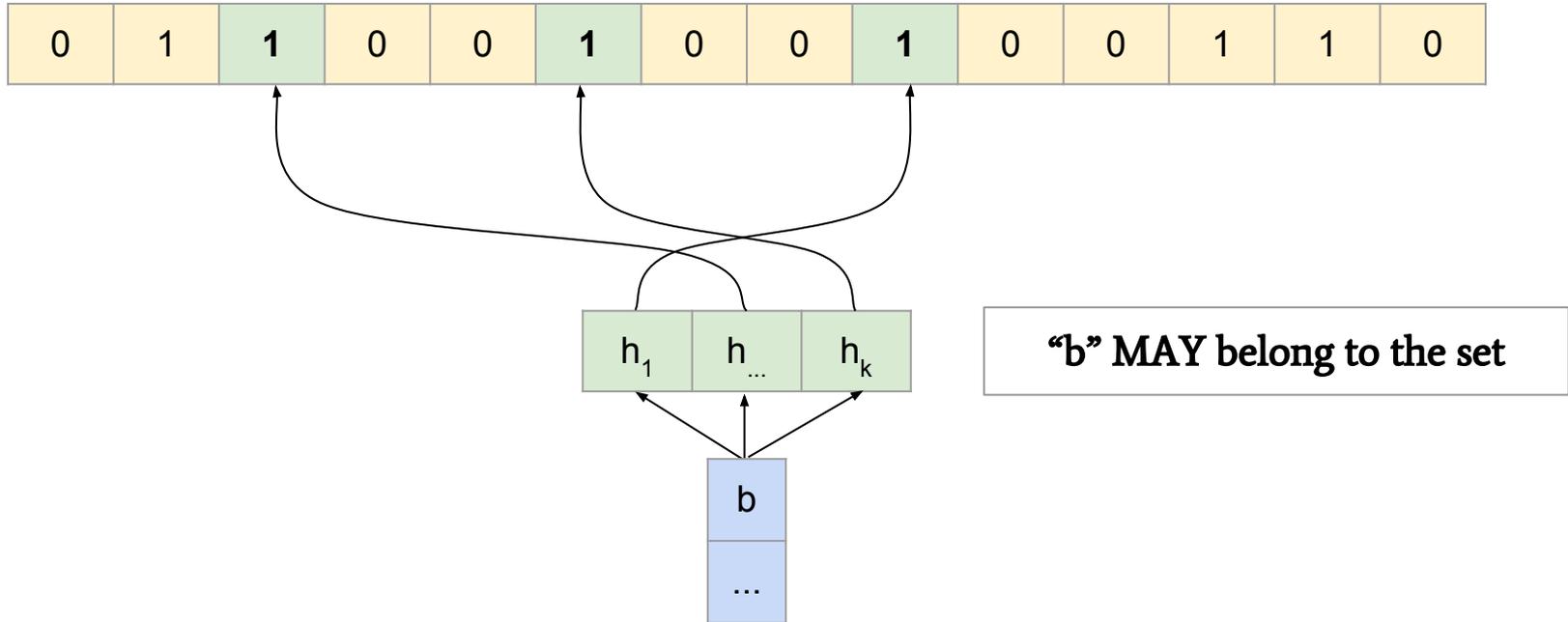
Bloom filter - Search [No false negative]



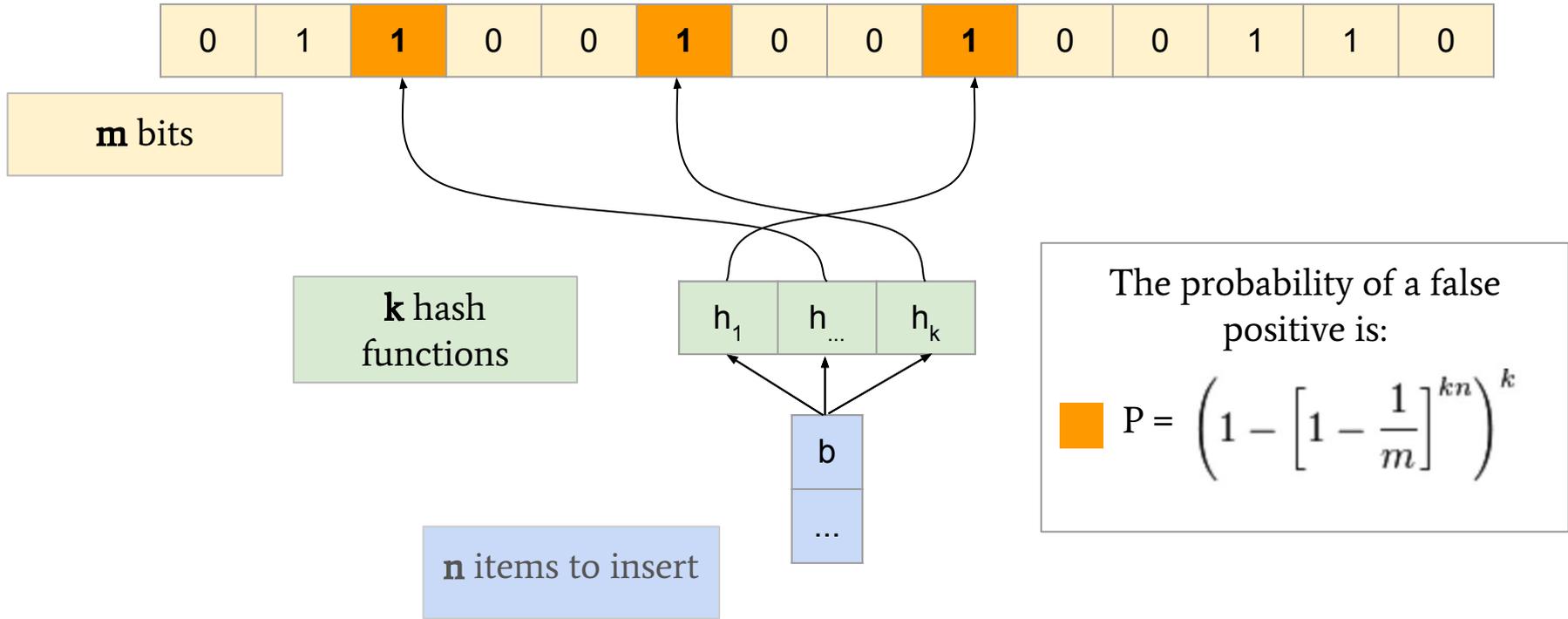
Bloom filter - Search [True positive]



Bloom filter - Search [Possible false positive]



Bloom filter - Analysis



The probability of a false positive is:

$$P = \left(1 - \left[1 - \frac{1}{m} \right]^{kn} \right)^k$$

Bloom filter - Implementation

m bits

k hash
functions

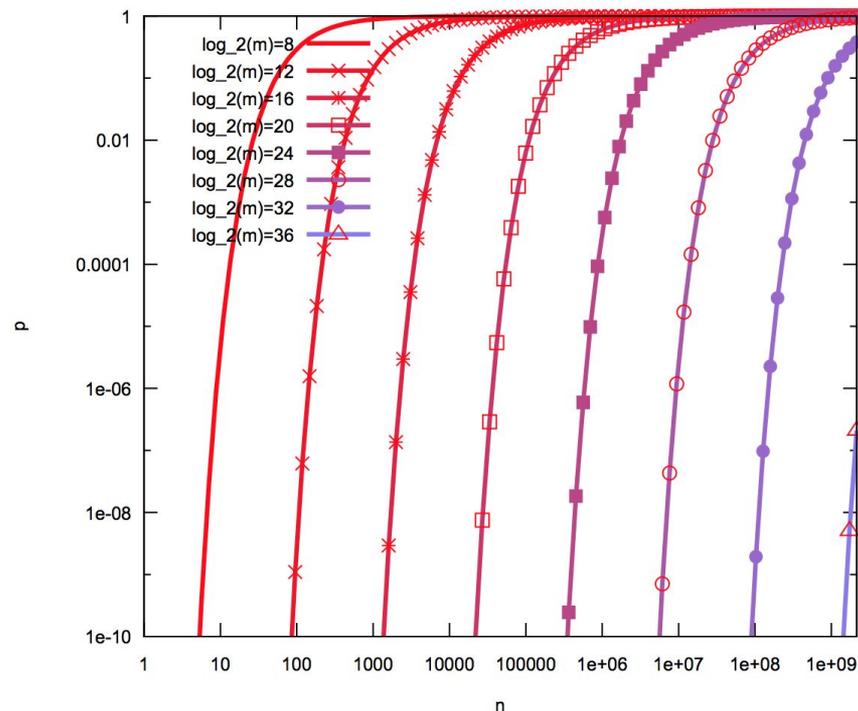
n items to insert

- Optimal number of hash function:

$$k = \frac{m}{n} \ln 2$$

- Optimal number of bit m for the desired probability p of false positive:

$$m = -\frac{n \ln p}{(\ln 2)^2}$$



Bloom filter - Results

Naïve approach

60+ GB

VS

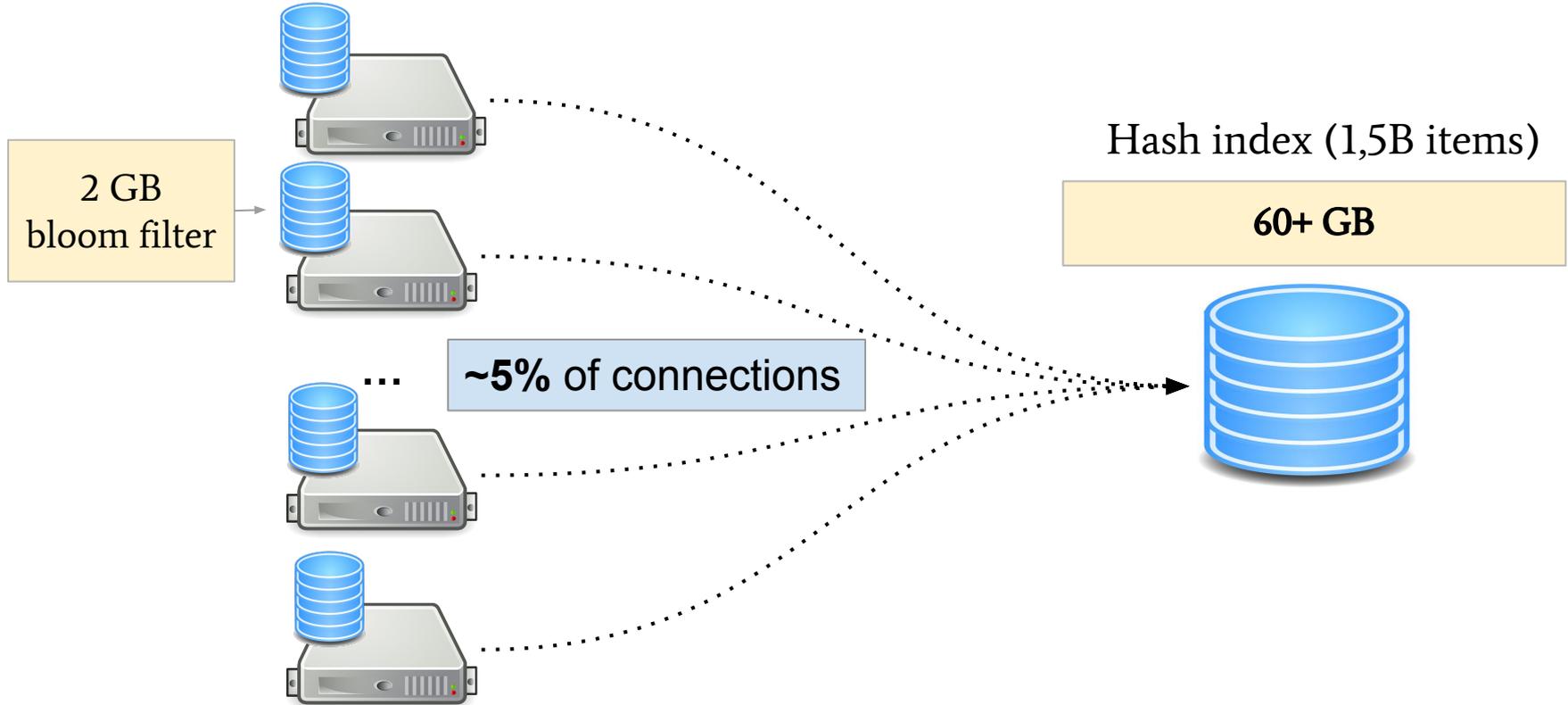
Bloom filter

2 GB (14B bit)

7 hash functions

1% of false positive

Bloom filter - Results [MyMemory]



Counting unique items in a stream



ClickMeter - Number of unique IPs per link

Flajolet - Martin for unique hash counting

Use case

The image shows a screenshot of the ClickMeter dashboard. The dashboard header includes the ClickMeter logo, navigation links for VIDEO, API, TOUR, PRICING, and LOGIN, and a main heading "Get the Best from your Links" with a sub-heading "Monitor, compare and optimize all your marketing links in one place to increase the conversion rate". Below this are buttons for "LEARN MORE" and "GET STARTED". The main content area displays a "List of tracking links" table with columns for Tracking link name, OSMA, UC, N/C, Competition, NOVA, and a dropdown menu for "View Landing page". The table lists various tracking links with their respective metrics and status indicators.

Target
visitors where they yield best conversion rates

Track
views, clicks and conversions

Monitor
broken links, click fraud, latency and blacklists

Share
with clients, partners, and coworkers

Counting unique elements could be really costly in terms of memory.

Counting unique items - Naïve approach

(4B bits array)



0.0.0.0

500 MB per link

255.255.255.255

5 PB with **10M** links

Counting unique items - Flajolet-Martin

Flajolet-Martin

...01010101001000

$P(n \text{ trailing zeros}) = ?$

Flajolet-Martin

...01010101001000

$$P(n \text{ trailing zeros}) = \left(\frac{1}{2}\right)^n$$

seen hashes \cong ?

Flajolet-Martin

...01010101001000

$$P(n \text{ trailing zeros}) = \left(\frac{1}{2}\right)^n$$

seen hashes $\cong 2^n$

... x x x x x x x x 000

... x x x x x x x x 001

... x x x x x x x x 010

... x x x x x x x x 011

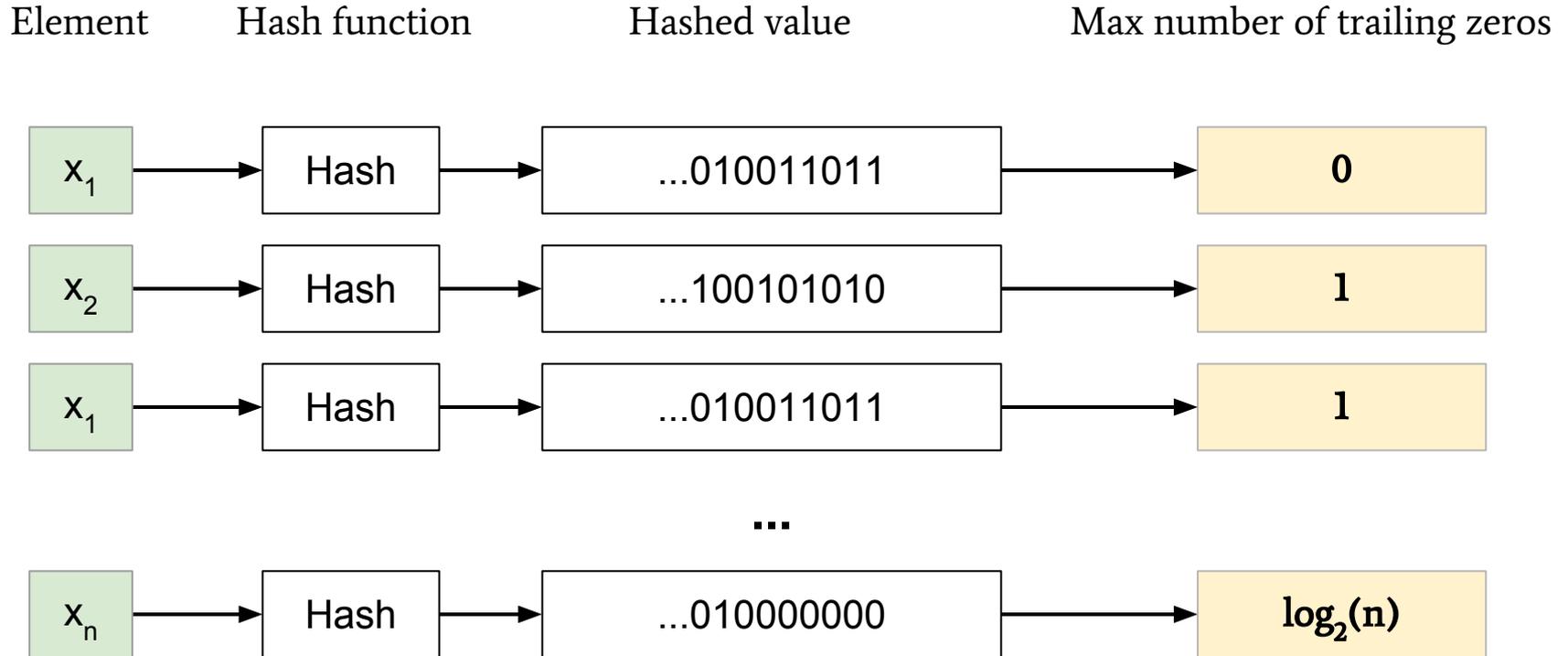
... x x x x x x x x 100

... x x x x x x x x 101

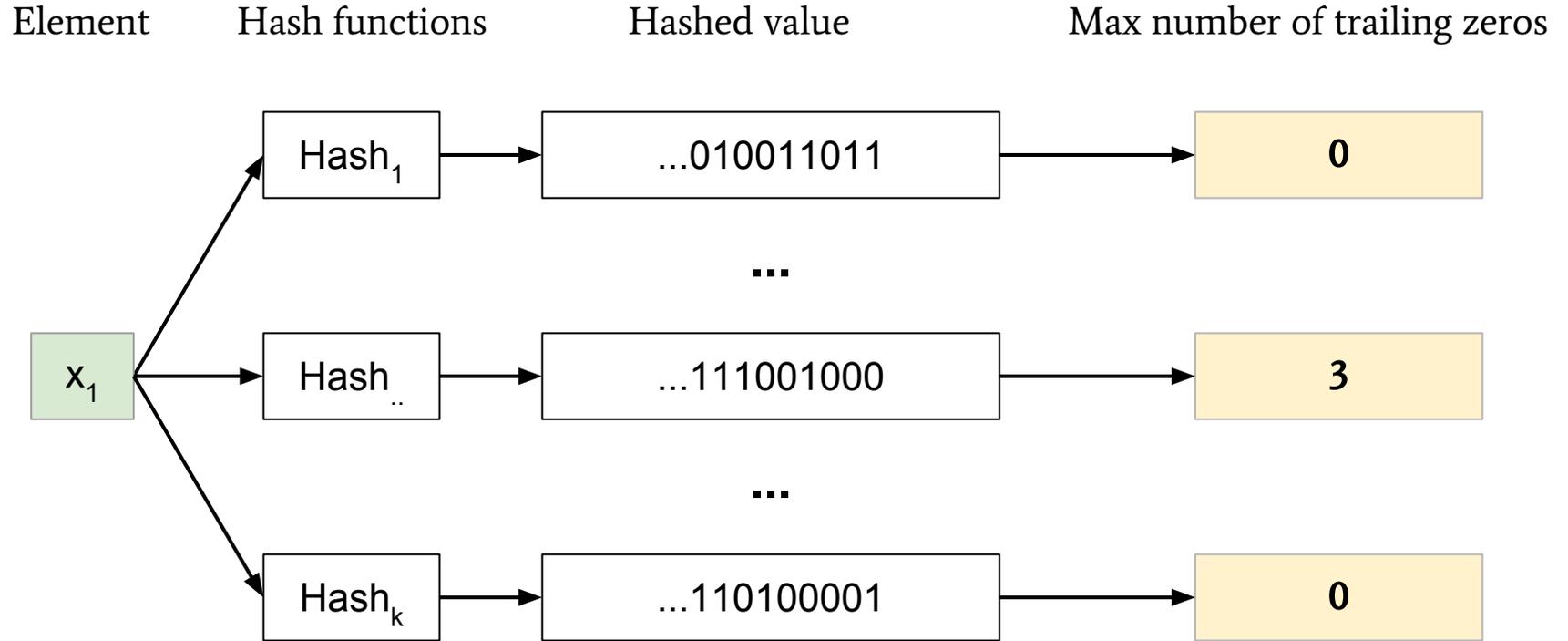
... x x x x x x x x 110

... x x x x x x x x 111

Flajolet-Martin



Flajolet-Martin



Flajolet-Martin - Results

Naïve approach

500 MB per link

5 PB with **10M** links

VS

Flajolet-Martin

1,5 KB per link

2% of error

15 GB with **10M** links

Probabilistic search



Memopal - Search for similar files

Local sensitive hashing & min hashing

Use case



Protect your files

With Memopal, you can back up all your files and free up space on your computer

DOWNLOAD FOR WINDOWS ⬇

Download Memopal and receive 3 GB for free. [All versions»](#)

The diagram illustrates the Memopal workflow: a smartphone on the left is connected via a Wi-Fi signal to a laptop in the center. The laptop screen shows a document with a yellow header, an orange section, and a green footer. A dashed line connects the laptop to a white cloud icon on the right, representing cloud storage synchronization.

The difference between a petabyte and a gigabyte index is worth an approximation.

Search - Naïve approach

2 B files

1 PB of index

Slow search

Search - Min hash

Similarity

Document 1

Midway upon the journey of
our life
I found myself within a forest
dark,
For the straightforward
pathway had been lost.
Ah me! how hard a thing it is
to say
What was this forest savage,
rough, and stern,
Which in the very thought
renews the fear.

Document 2

Day was departing, and the
embrowned air
Released the animals that are
on earth
From their fatigues; and I the
only one
Made myself ready to sustain
the war,
Both of the way and likewise
of the woe,
Which memory that errs not
shall retrace.

Are they similar?

$$\text{Jaccard} = \frac{\text{Number of substrings in common}}{\text{Total number of unique substrings}}$$

Similarity

Substrings => Shingles of length S

“Midway upon the journey of our life”

Set of shingles = $\left[\begin{array}{c} \dots \\ \text{“Midway upon the”}, \\ \text{“upon the journey”}, \\ \text{“the journey of”}, \\ \dots \end{array} \right]$

Storage \cong S * Doc_length * #Docs
Complexity \cong Doc_length * #Docs

Similarity

Fingerprint => 32 bit hash of a shingle

Set of shingles = $\left[\begin{array}{c} \dots \\ \dots 100101101 \dots, \\ \dots 011010000\dots, \\ \dots 110010011 \dots, \\ \dots \end{array} \right]$

Storage \cong **4 byte** * Doc_length * #Docs
Complexity \cong Doc_length * #Docs

Similarity

We need to find a signature $\text{Sig}(D)$ of length K so that

if $\text{Sig}(D_1) \sim \text{Sig}(D_2)$ then $D_1 \sim D_2$

Storage \cong 4 byte * K * #Docs
Complexity \cong K * #Docs

With $K \ll \text{Doc_length}$

MinHash - Signature creation

Generate the fingerprints
of the documents.

Doc ₁
...10101
...01100
...10010
...00111

H_n

Take a random permutation
of the fingerprints.

Doc ₁
...00111
...01100
...10101
...10010

*Minhash of this
permutation*

Define $\text{minhash}(H_n, \text{Doc}_i) = \text{First fingerprint of Doc}_i \text{ hashed with } H_n$

$\text{Sig}(\text{Doc}_i)$ of length $K = [\text{minhash}_1, \text{minhash}_2, \dots, \text{minhash}_n]$

MinHash

$\text{Sig}(\text{Doc})$ is a set of K min-hashing fingerprints:

Signature(Doc_1)

$$\left[\begin{array}{c} \dots 100101101 \dots \\ \dots 011010000 \dots \\ \dots 110010011 \dots \\ \dots 011100011 \dots \\ \dots 100100001 \dots \\ \dots \end{array} \right]$$

...

Signature(Doc_n)

$$\left[\begin{array}{c} \dots 100001101 \dots \\ \dots 101010110 \dots \\ \dots 110010011 \dots \\ \dots 010100101 \dots \\ \dots 100100001 \dots \\ \dots \end{array} \right]$$

MinHash

If $\text{Sig}(D_1) \sim \text{Sig}(D_2)$ then $\text{Doc}_1 \sim \text{Doc}_2$

Signature(Doc_1)	X	Signature(Doc_2)
... 100101101 ...	1	... 100001101 ...
... 011010000 ...	0	... 101010110 ...
... 110010011 ...	1	... 110010011 ...
... 011100011 ...	0	... 010100101 ...
... 100100001 ...	1	... 100100001 ...
...

$$P(X = 1) = \text{Jaccard}(\text{Doc}_1, \text{Doc}_2)$$

$$\sum X / K \approx \text{Jaccard}(\text{Doc}_1, \text{Doc}_2)$$

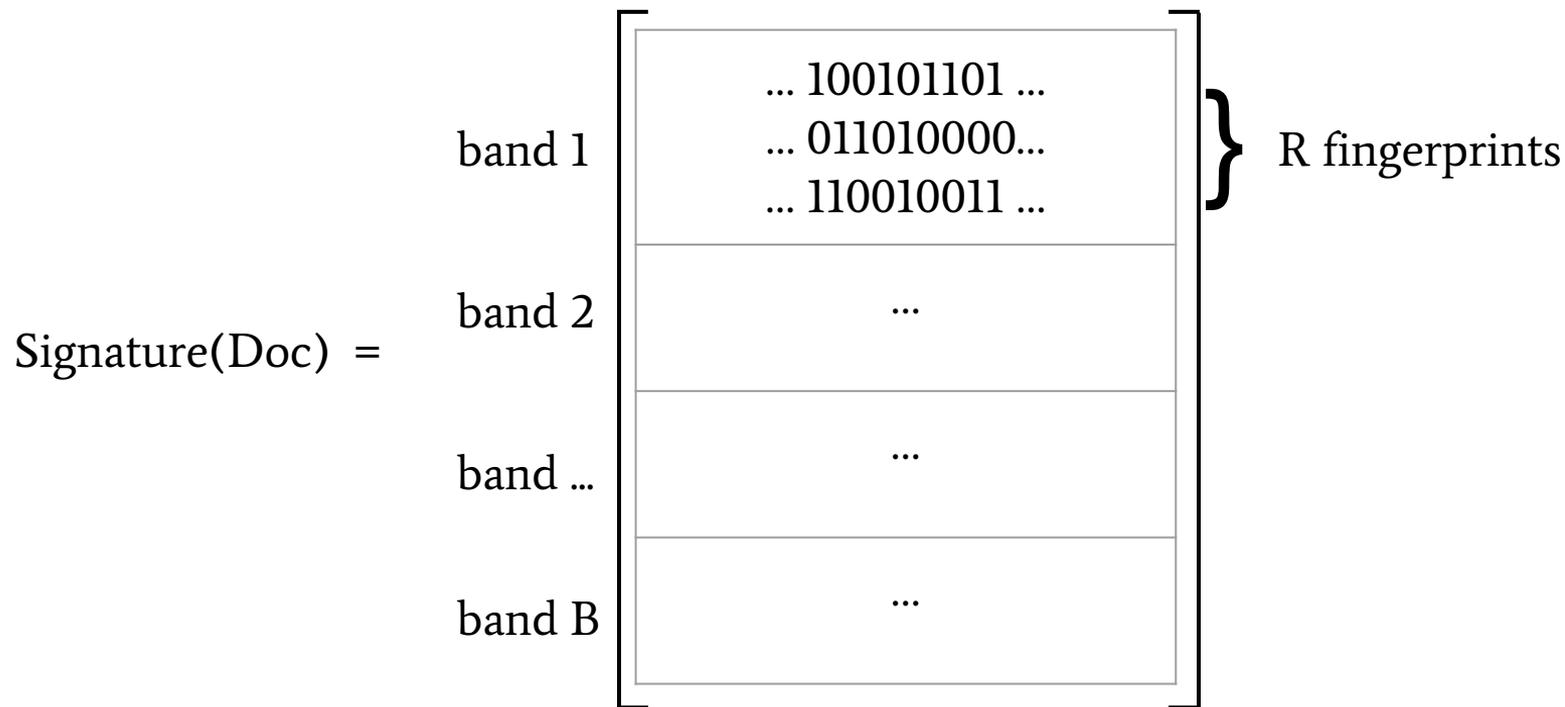
MinHash - Implementation

1. Generate the fingerprints of the document
2. Define K hash functions: h_1, h_2, \dots, h_K .
3. Define $\text{Sig}(\text{Doc}) = [h_1(\text{Doc}), h_2(\text{Doc}), \dots, h_K(\text{Doc})]$
4. Define $O = \{ i / h_i(\text{Doc}_1) = h_i(\text{Doc}_2) \}$
5. $\text{Sim}(\text{Doc}_1, \text{Doc}_2) = \frac{|O|}{K} \approx \text{Jaccard}(\text{Doc}_1, \text{Doc}_2)$

Storage \cong 4 byte * K * #Docs With $K \ll \text{Doc_length}$
Complexity \cong K * #Docs

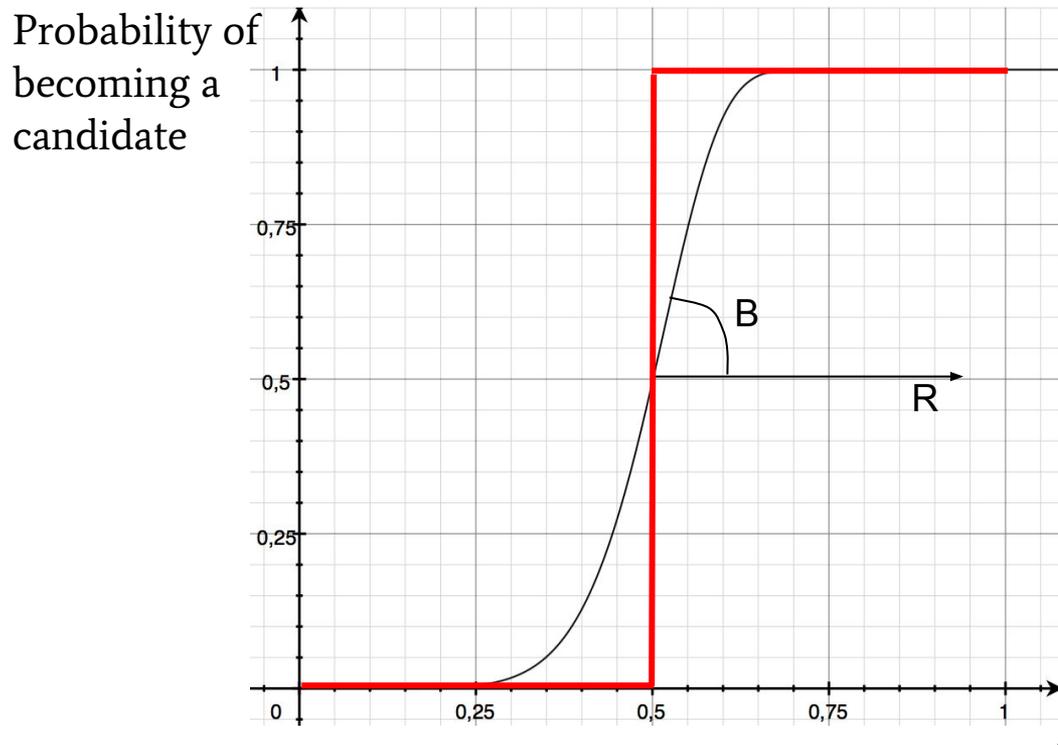
Local Sensitive Hashing

Divide the signature $\text{Sig}(\text{Doc})$ into **B bands of R rows each**, such that $B \cdot R = K$:



Local Sensitive Hashing - Analysis

Probability of a document having **at least band in common**: $1 - (1 - j^R)^B$

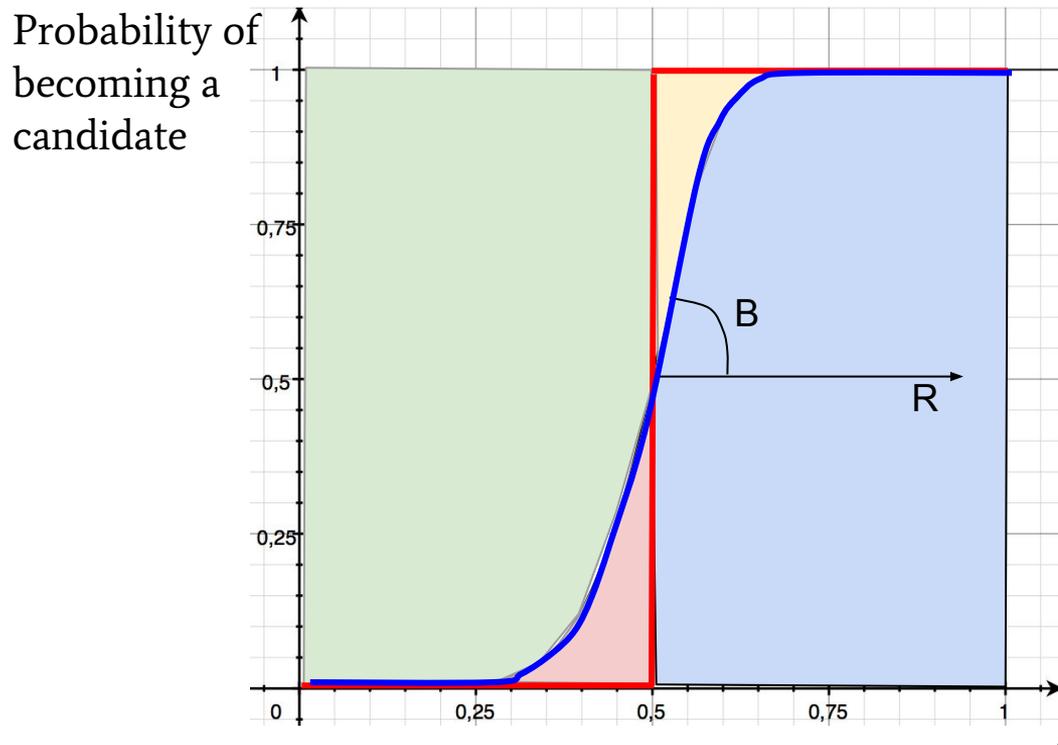


S-curve

- **Threshold** $\cong (1/B)^{1/R}$

Local Sensitive Hashing - Analysis

Probability of a document having **at least band in common**: $1 - (1 - j^R)^B$



S-curve

- **Threshold** $\cong (1/B)^{(1/R)}$
- True Positive
- True Negative
- False Positive
- False Negative

Probabilistic search - Results

From:

$$\begin{aligned}\text{Storage} &\cong \text{Shingle_length} * \text{Doc_length} * \#\text{Docs} \\ \text{Complexity} &\cong \text{Doc_length} * \#\text{Docs}\end{aligned}$$

To:

$$\begin{aligned}\text{Storage} &\cong 4 \text{ byte} * K * \#\text{Docs} \\ \text{Complexity} &\cong K * \#\text{Docs} * p(\text{“candidate”})\end{aligned}$$

With $K \ll \text{Doc_length}$ and $p(\text{“candidate”}) \ll 1$

Probabilistic search - Results

Naïve approach

2 B files

1 PB of index

Slow search

VS

Min hash + LSH

2 B files

1,5 TB of index

Fast search & update

Thank you

...

$$P(|\text{questions}| > 0) = 1 - [1 - p(\text{question})]^{|\text{audience}|}$$

...

Any questions?