Hash

...

A probabilistic approach for big data

Luca Mastrostefano

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

...

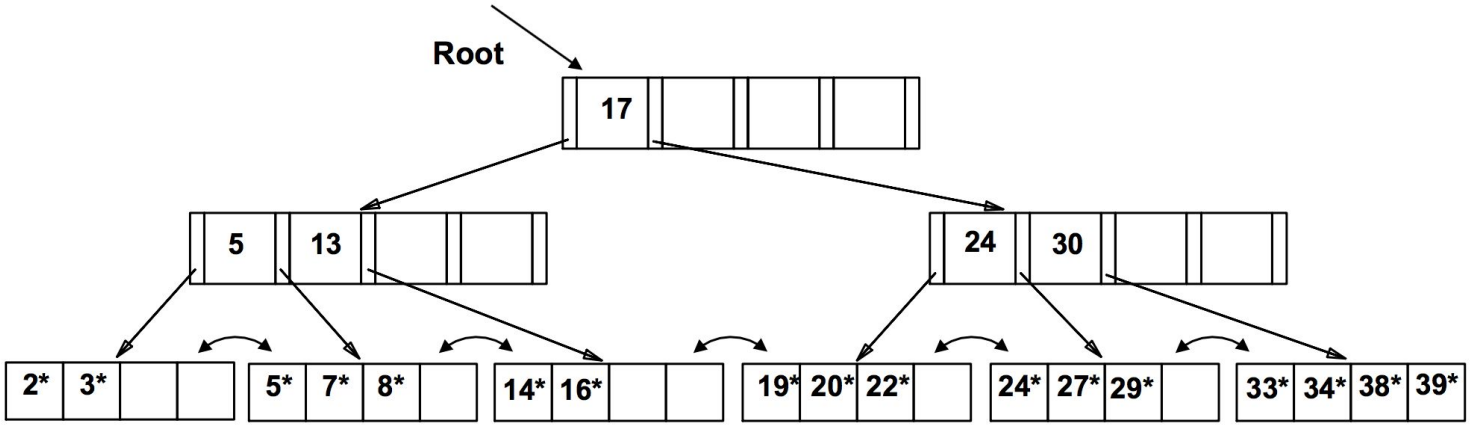
Databases - Fast and exact search

Static, extendible and linear hash indexes

Use case

Sometimes also a logarithmic complexity is too expensive.

B^+ tree index

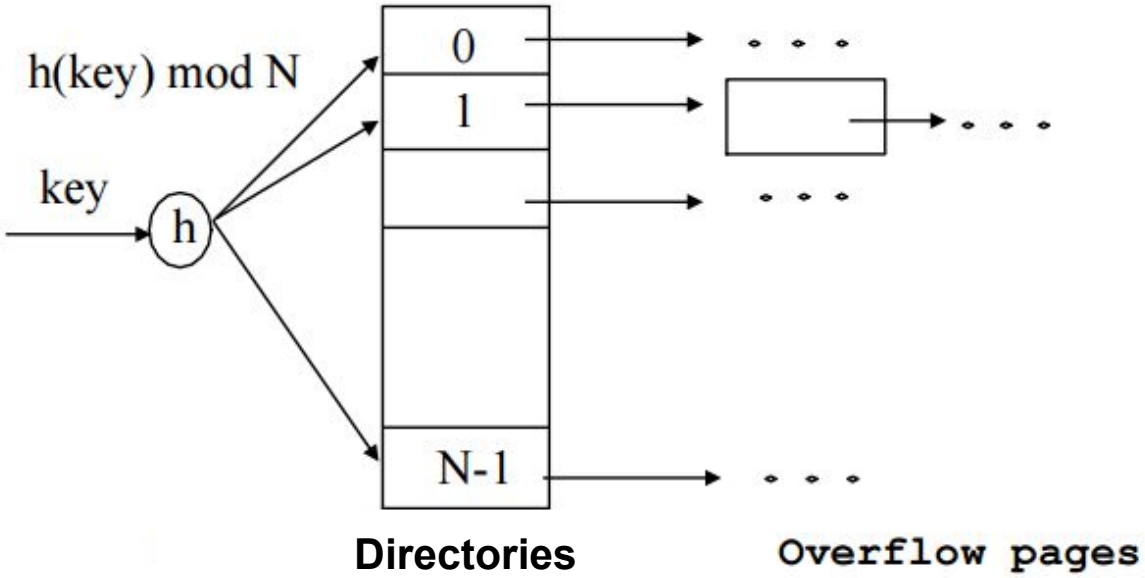


Select/Insert ≅ Log_F(# items)

Images from Data Management - Maurizio Lenzerini

Search - Hash index

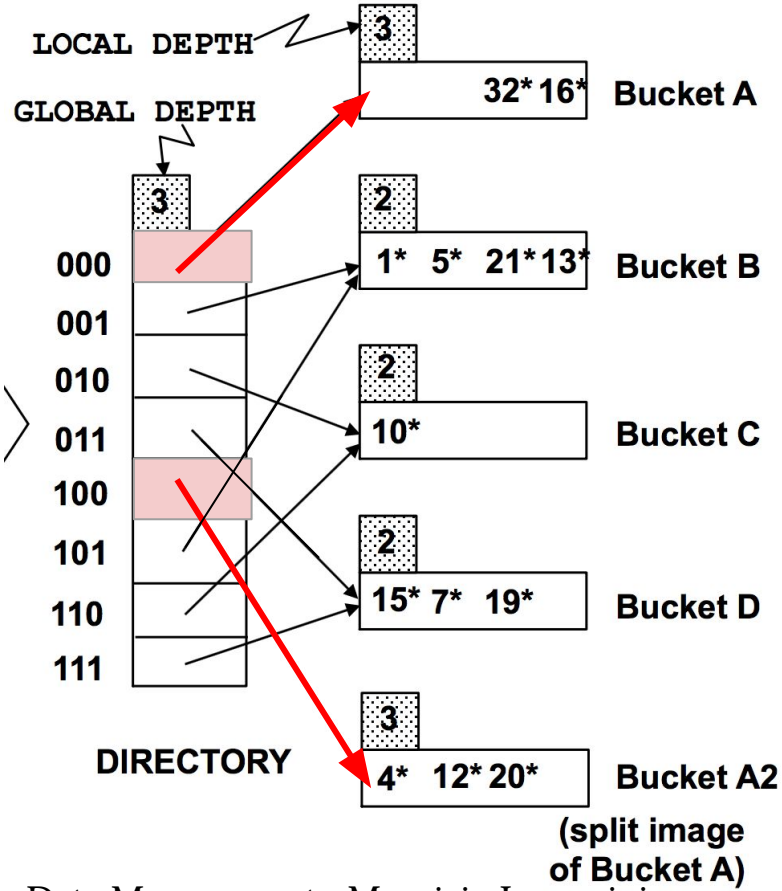
Static hash index



Select/Insert ≅ 2 + (# overflow pages)

Images from Data Management - Maurizio Lenzerini

Dynamic hash index - Extendible



Select/Insert ≅
2 + (# overflow pages)

 # overflow pages almost constant

Images from Data Management - Maurizio Lenzerini

Dynamic hash index - Linear

Intuition:

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

Select/Insert ≅
1 + (# overflow pages)

 # overflow pages almost constant

Indexes comparison - Secondary memory accesses

B^+ tree index	VS	Linear hash index
Select/Insert ≃ Log		Select/Insert ≃ const
4 accesses ≃ 30 ms		**1 access ≃ 7 ms**

4x in case of billions of entries

Stream filter: $x \in U$?

...

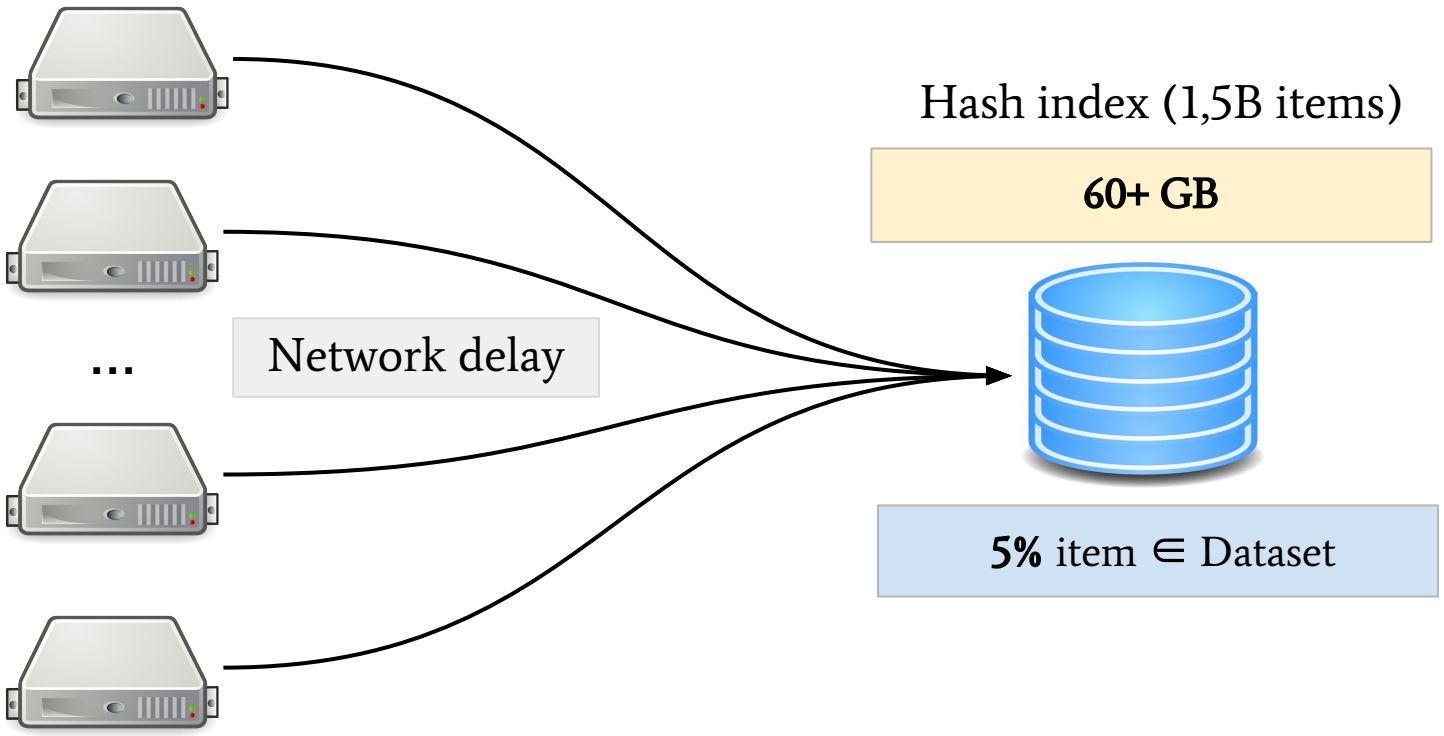
Translated - MyMemory

Bloom filter

Use case

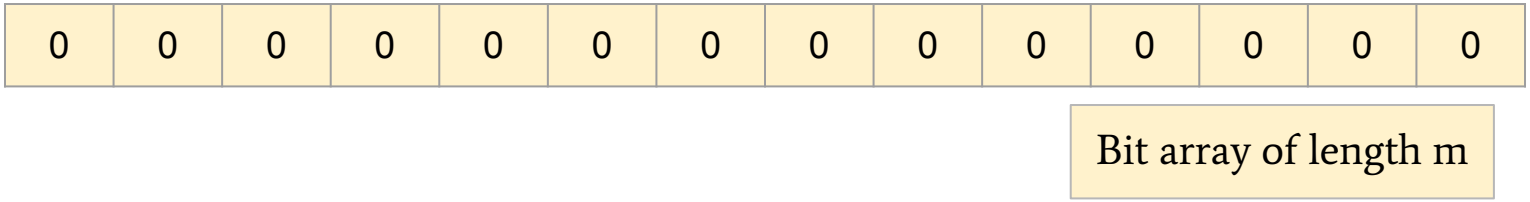
The delay introduced by the secondary memory does not fit an environment in which milliseconds matter.

Stream filter - Naïve approach

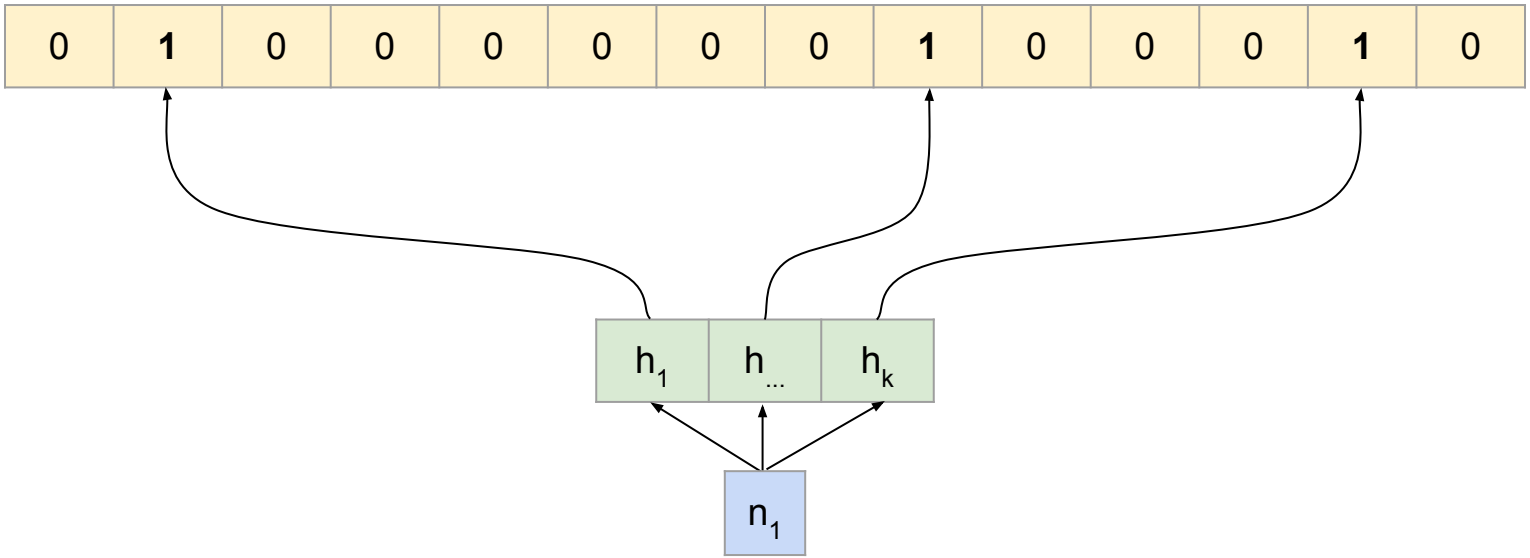


Stream filter - Bloom filter

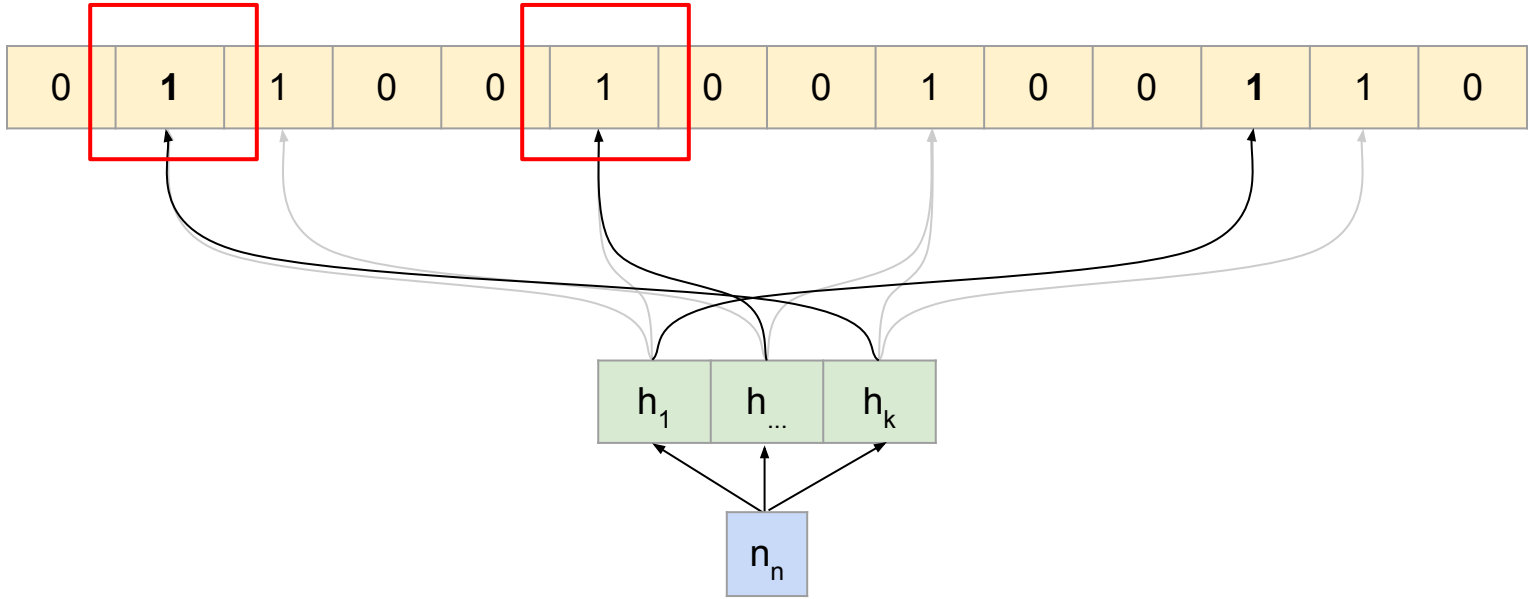
Bloom filter - Insert



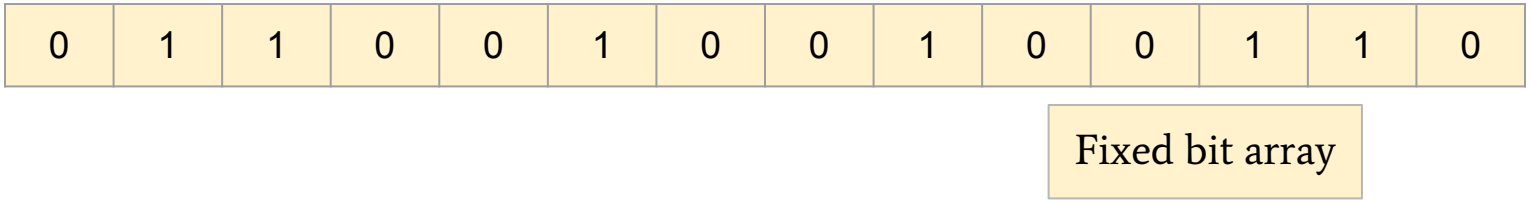
Bloom filter - Insert



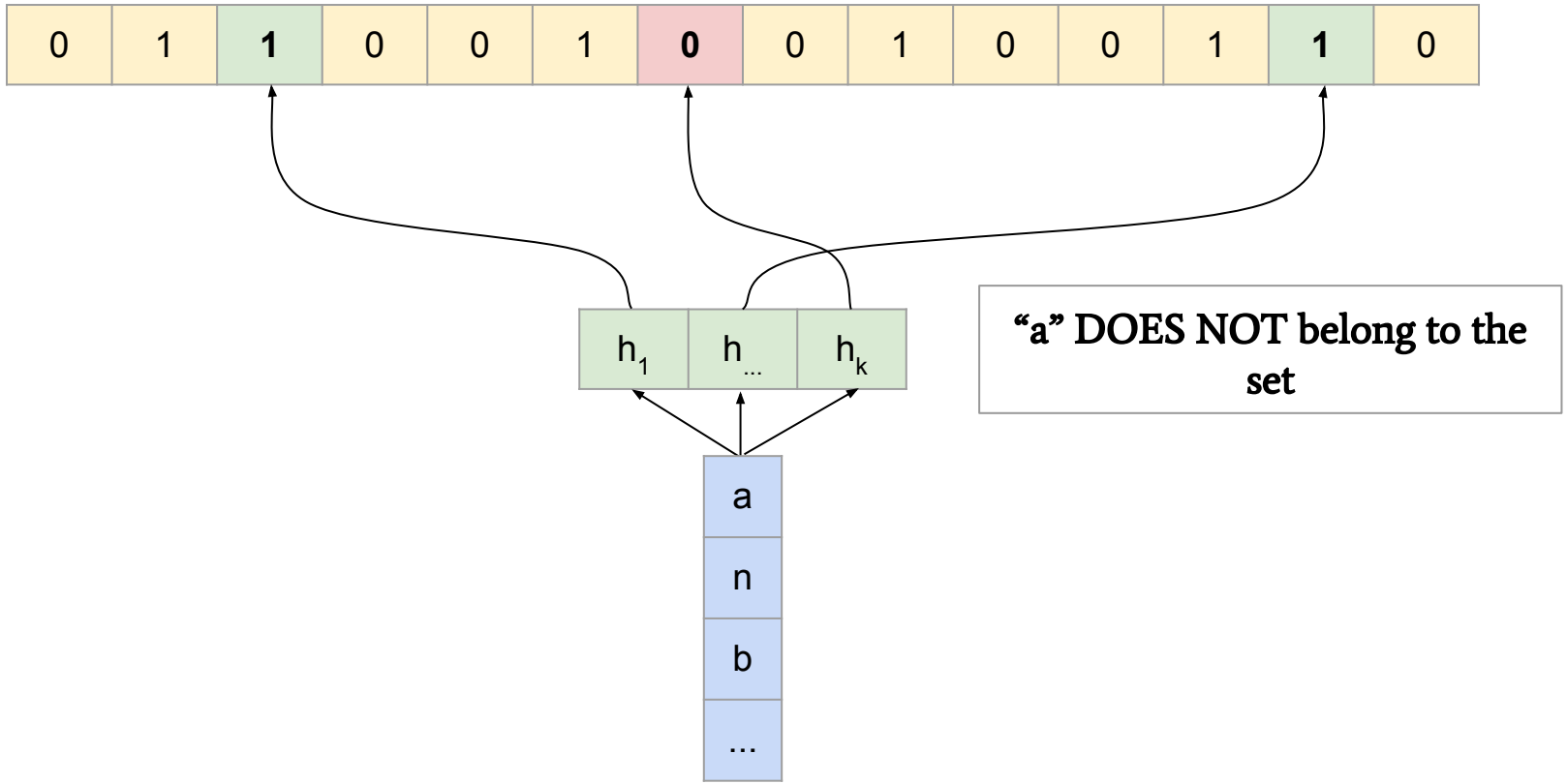
Bloom filter - Insert



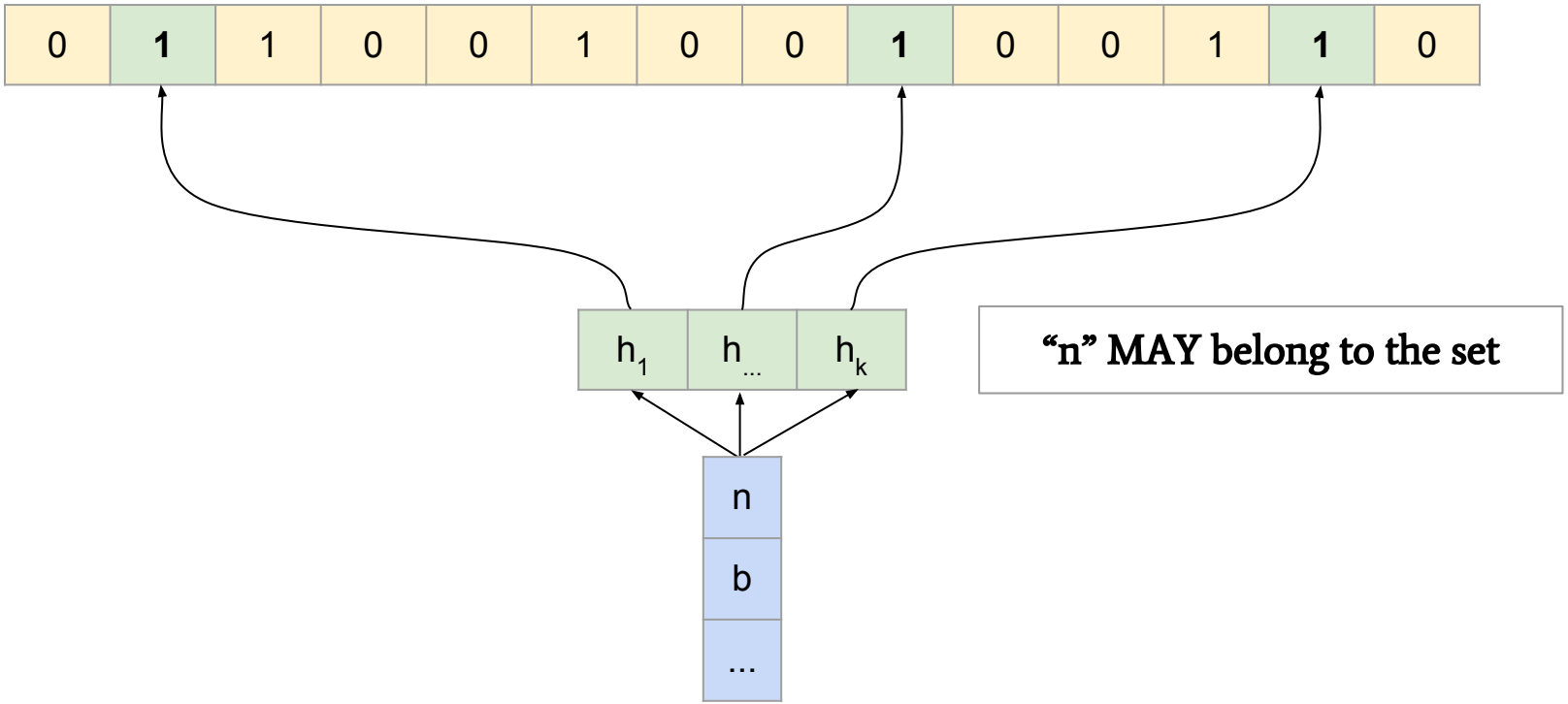
Bloom filter - Search



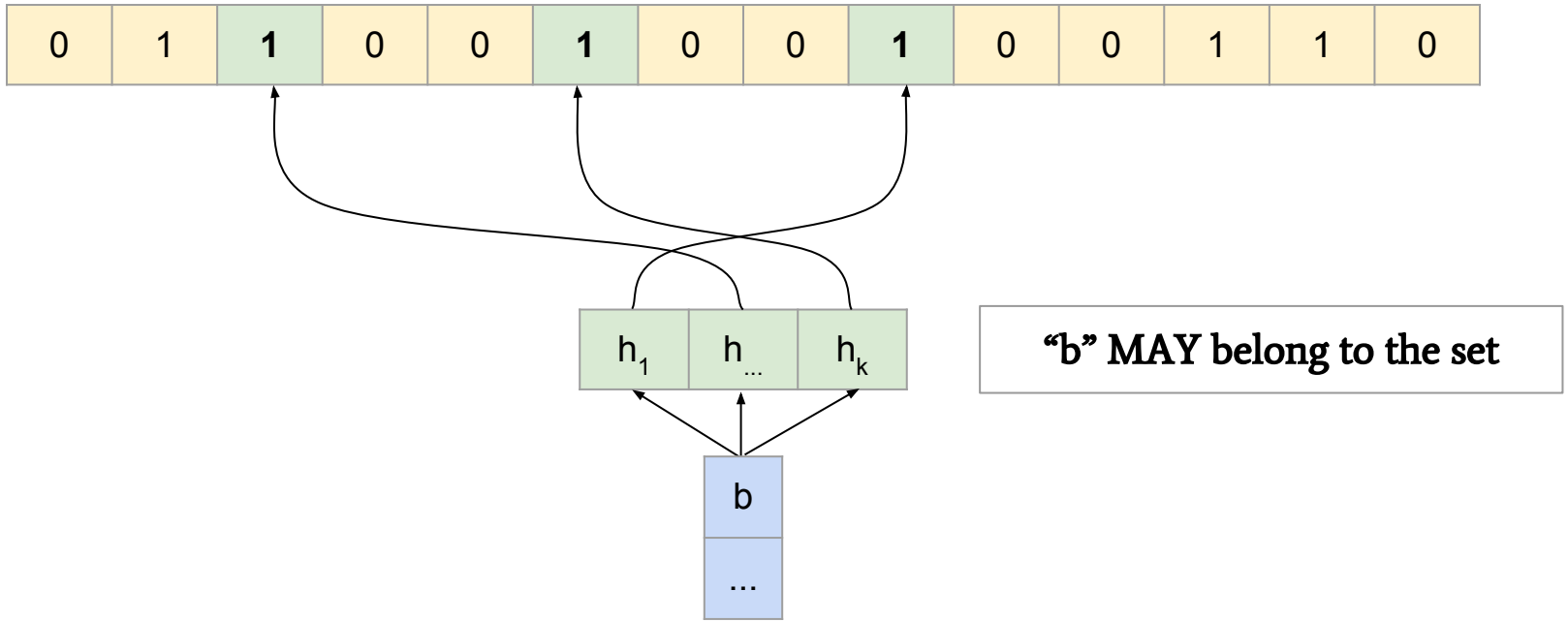
Bloom filter - Search [No false negative]



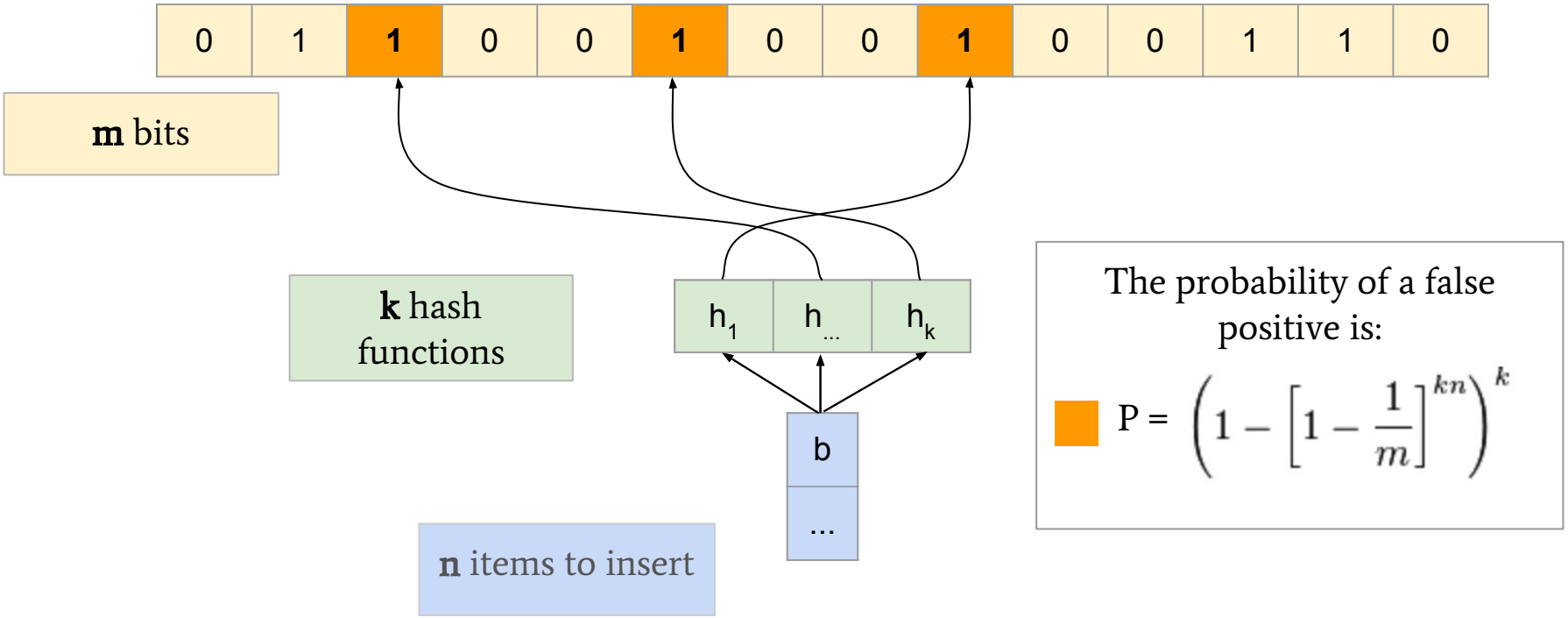
Bloom filter - Search [True positive]



Bloom filter - Search [Possible false positive]



Bloom filter - Analysis



Bloom filter - Implementation

m bits

k hash functions

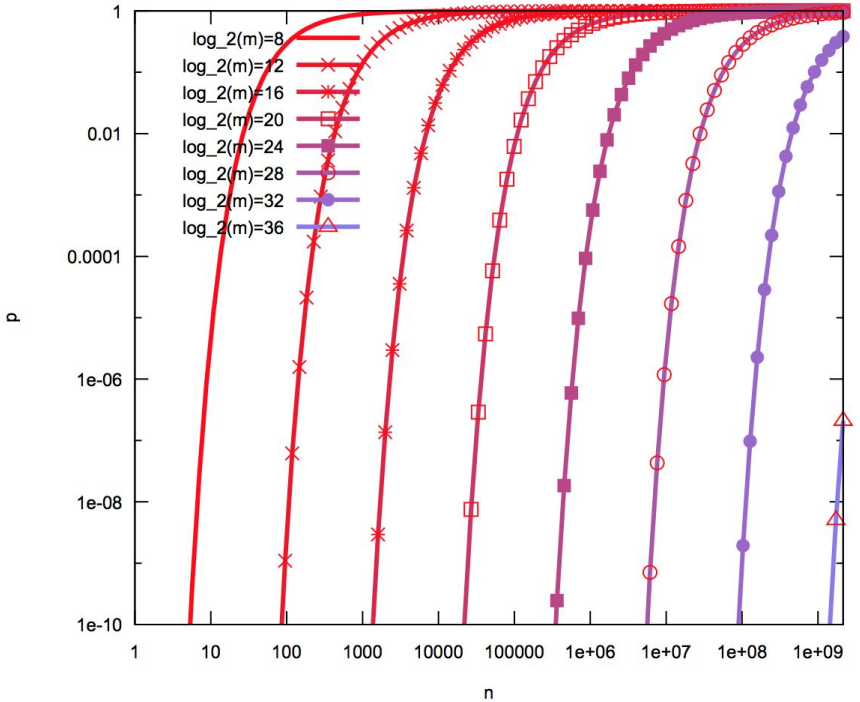
n items to insert

- 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

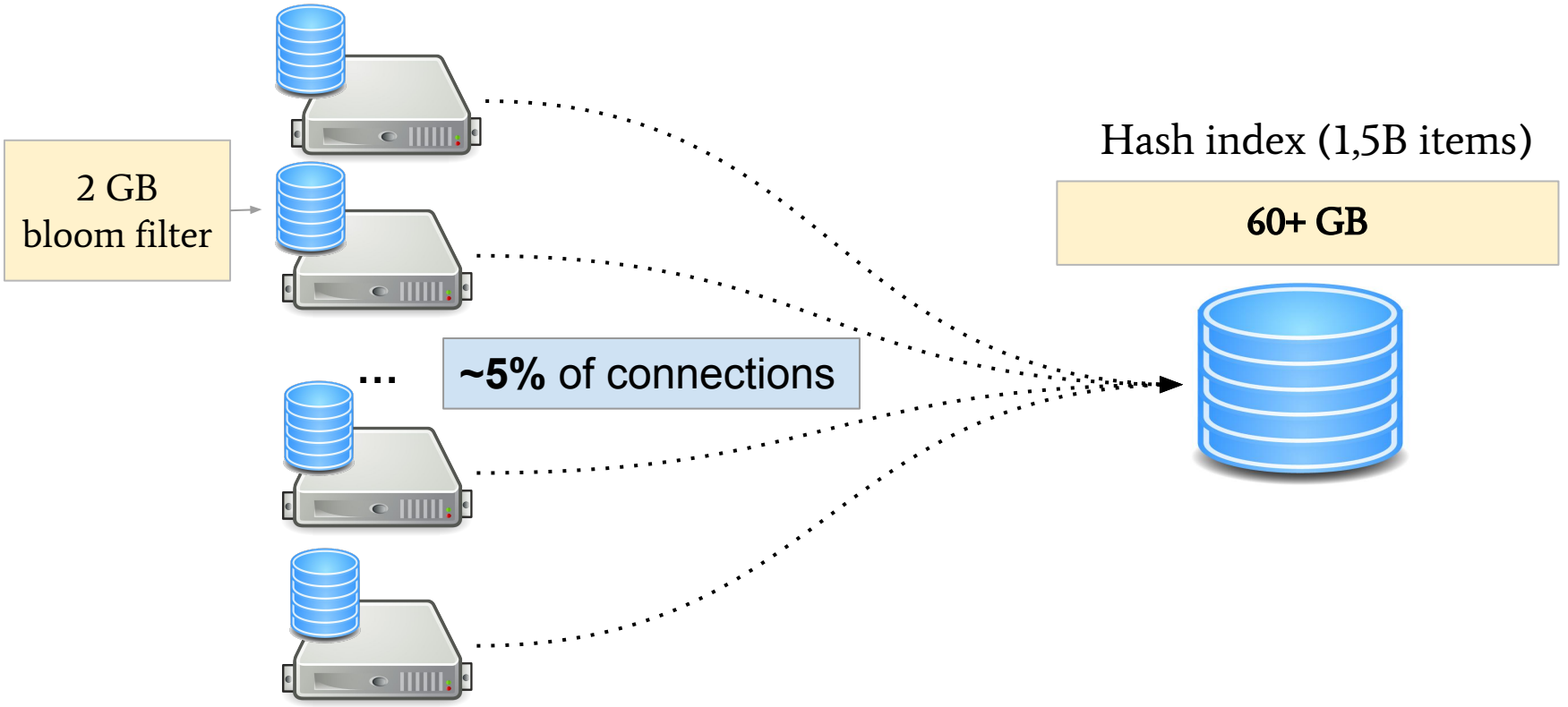
Bloom filter

2 GB (14B bit)

7 hash functions

1% of false positive

Bloom filter - Results [MyMemory]



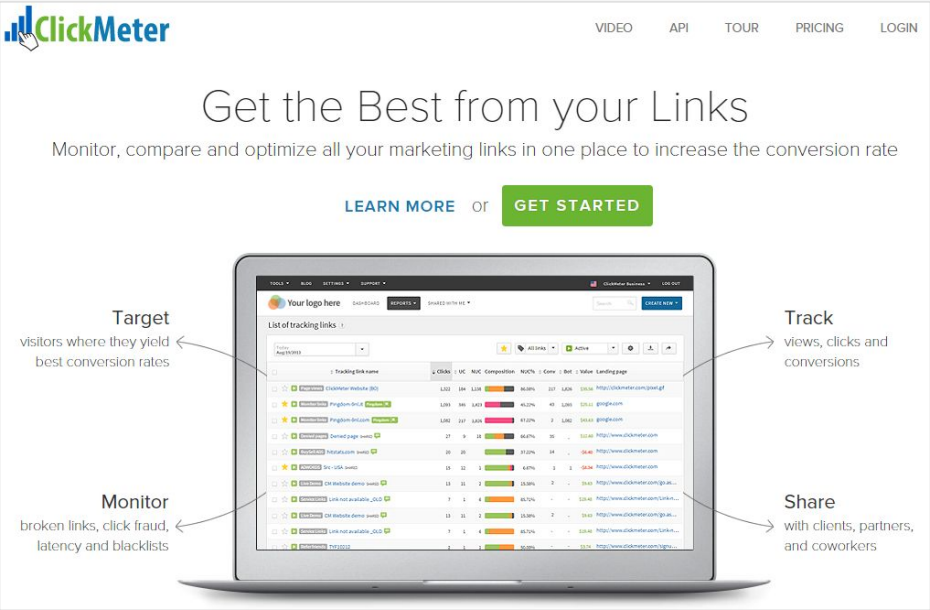
Counting unique items in a stream

...

ClickMeter - Number of unique IPs per link

Flajolet - Martin for unique hash counting

Use case



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

...0 1 0 1 0 1 0 1 0 0 1 0 0 0

P(n trailing zeros) = ?

Flajolet-Martin

...0 1 0 1 0 1 0 1 0 0 1 0 0 0

P(n trailing zeros) = (½)^n

seen hashes ≅ ?

Flajolet-Martin

...0 1 0 1 0 1 0 1 0 0 1 0 0 0

P(n trailing zeros) = (½)^n

seen hashes ≅ **2^n**

... x x x x x x x x 0 0 0

... x x x x x x x x 0 0 1

... x x x x x x x x 0 1 0

... x x x x x x x x 0 1 1

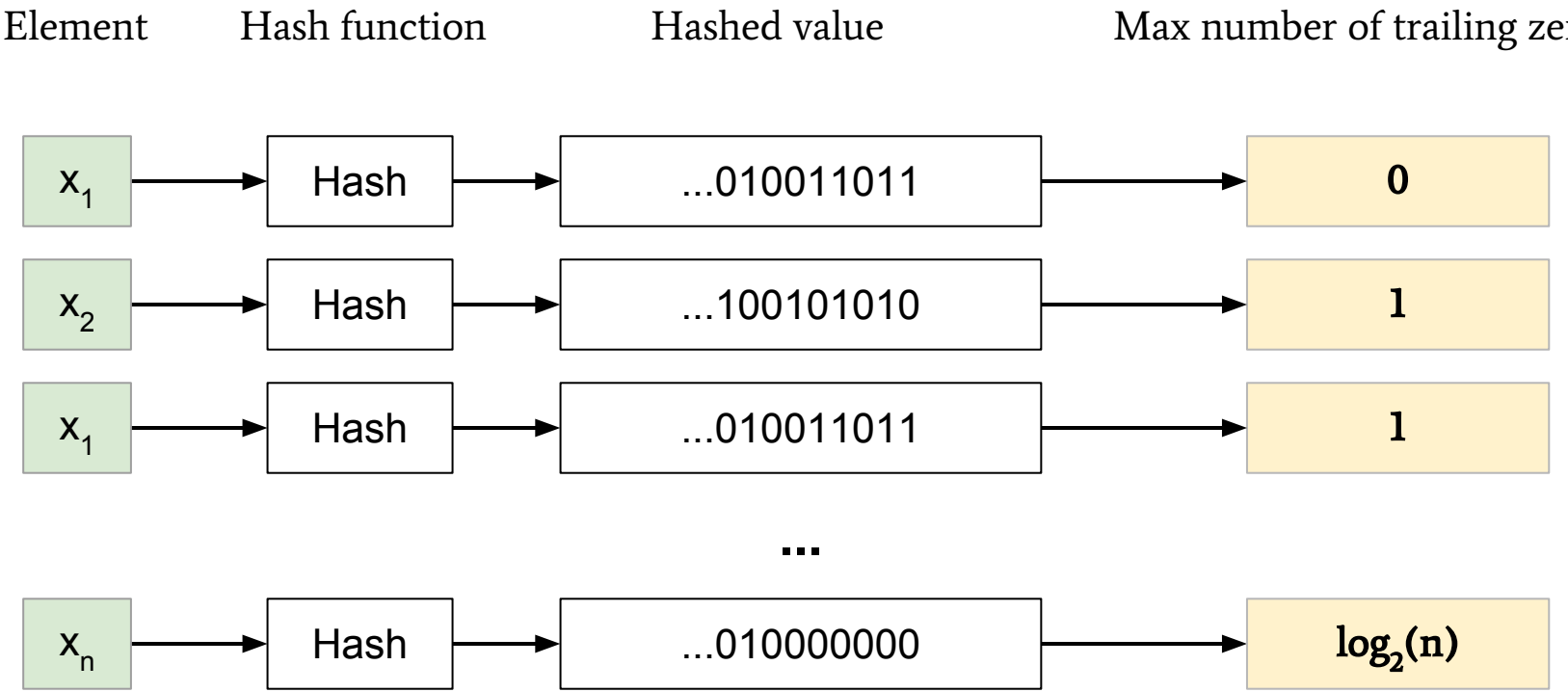
... x x x x x x x x 1 0 0

... x x x x x x x x 1 0 1

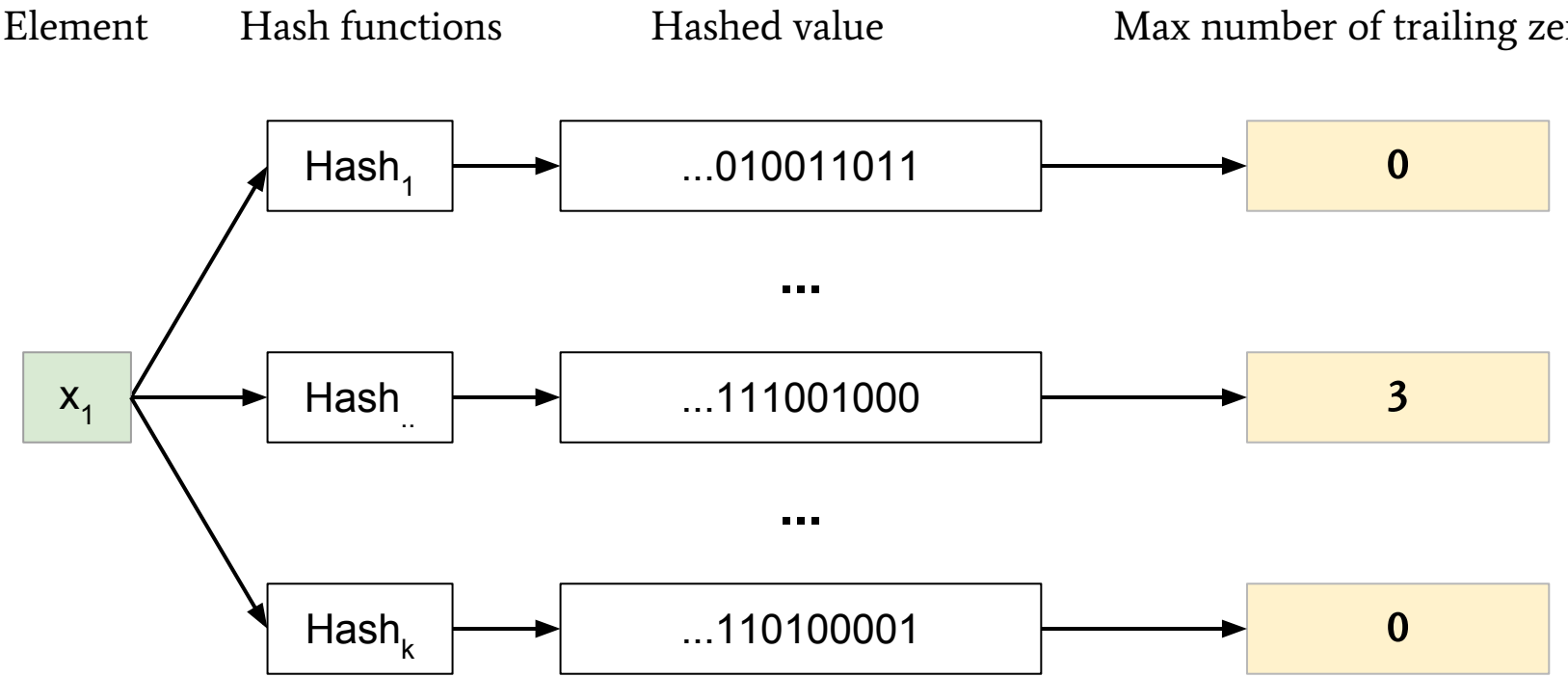
... x x x x x x x x 1 1 0

... x x x x x x x x 1 1 1

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

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.

Are they similar?

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.

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

Similarity

Substrings => Shingles of length S

“Midway upon the journey of our life”

Set of shingles = [… “Midway upon the”, “upon the journey”, “the journey of”, …]

Storage	≅	S * Doc_length * #Docs
Complexity	≅	Doc_length * #Docs

Similarity

Fingerprint => 32 bit hash of a shingle

Set of shingles = [... ... 100101101 ..., ... 011010000..., ... 110010011 ..., ...]

Storage	≅	**4 byte** * Doc_length * #Docs
Complexity	≅	Doc_length * #Docs

Similarity

We need to find a signature Sig(D) of length K so that

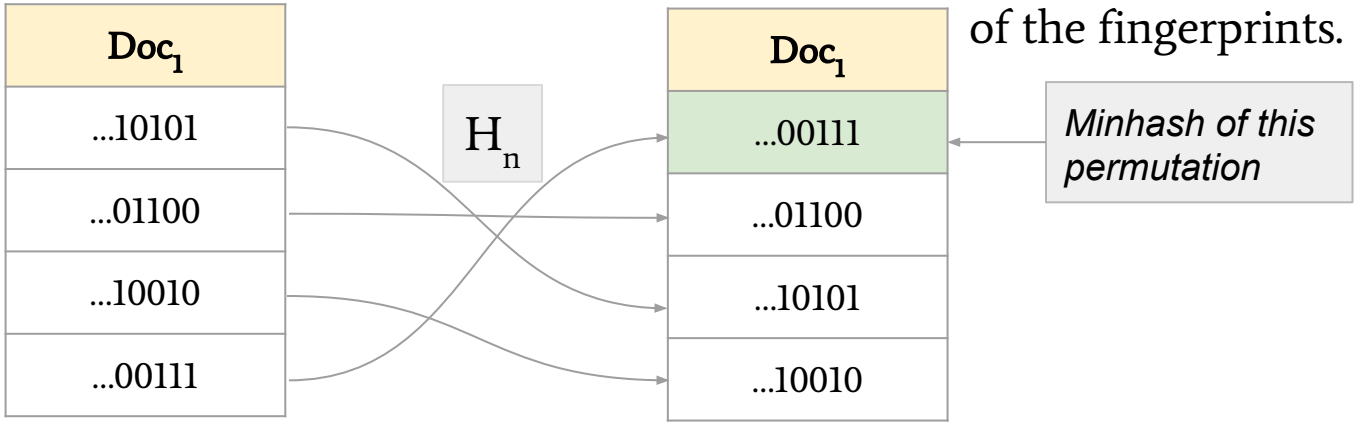
if $Sig(D_1) \sim Sig(D_2)$ then $D_1 \sim D_2$

Storage	≅	4 byte * **K** * #Docs	With K << Doc_length
Complexity	≅	**K** * #Docs	

MinHash - Signature creation

Generate the fingerprints of the documents.

Take a random permutation of the fingerprints.

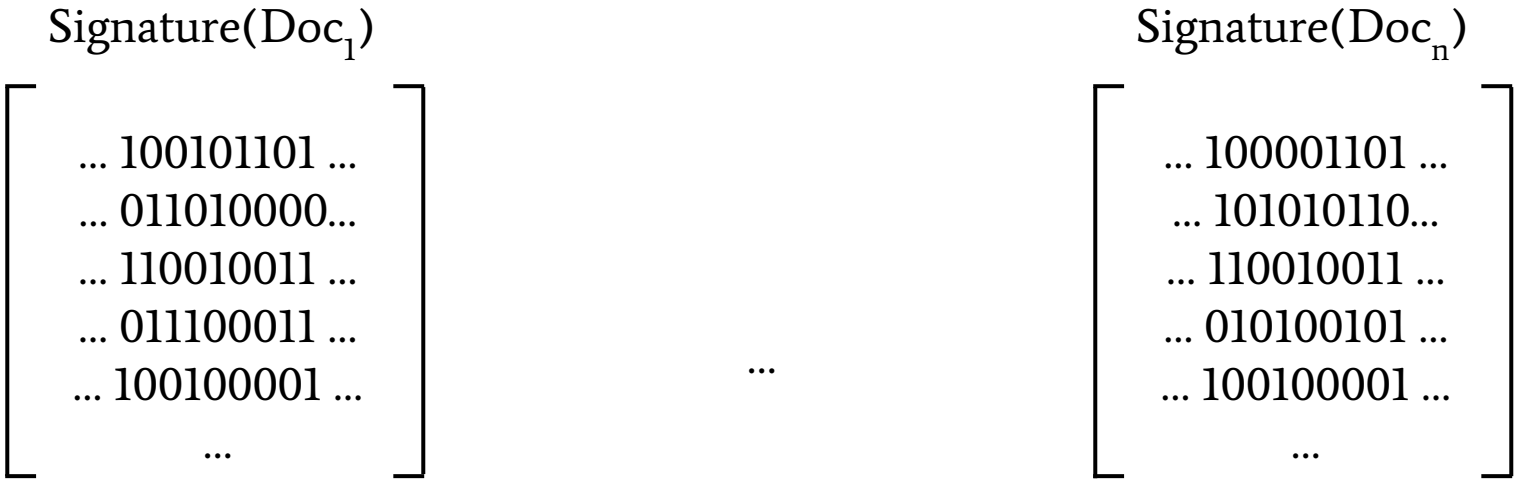


Define minhash(H_n, Doc_i) = First fingerprint of Doc_i hashed with H_n

Sig(Doc_i) of length K = [$minhash_i$, $minhash_2$, ..., $minhash_n$]

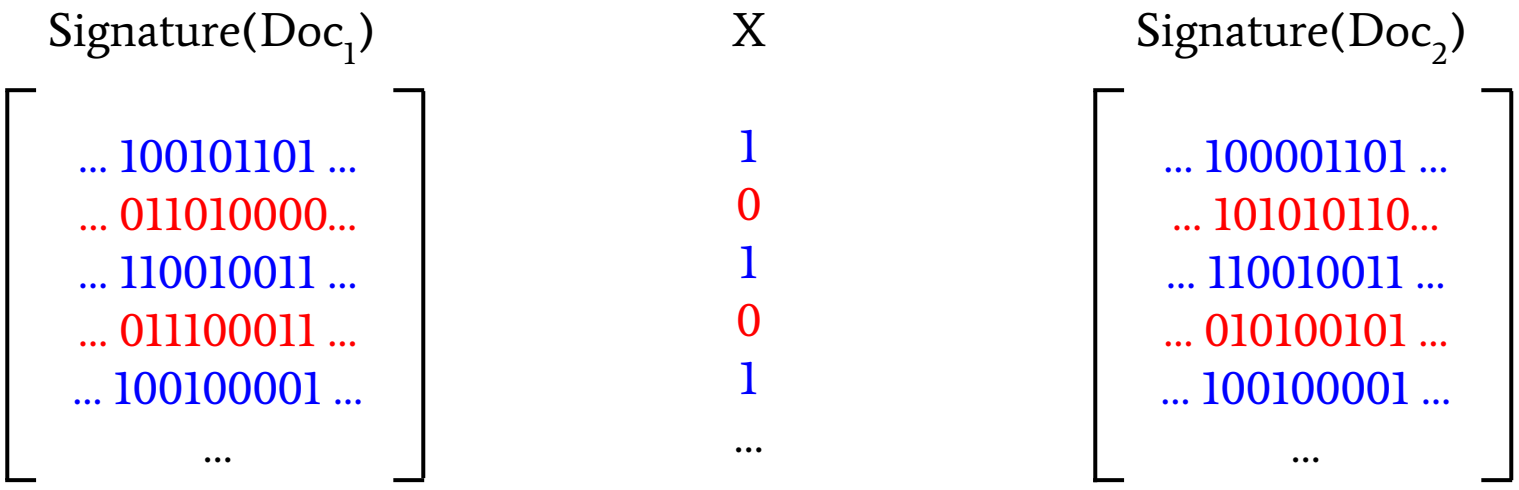
MinHash

Sig(Doc) is a set of K min-hashing fingerprints:



MinHash

If $Sig(D_1) \sim Sig(D_2)$ then $Doc_1 \sim Doc_2$



$$P(X = 1) = Jaccard(Doc_1, Doc_2)$$

$$\sum X / K \simeq \mathbf{Jaccard(Doc_1, Doc_2)}$$

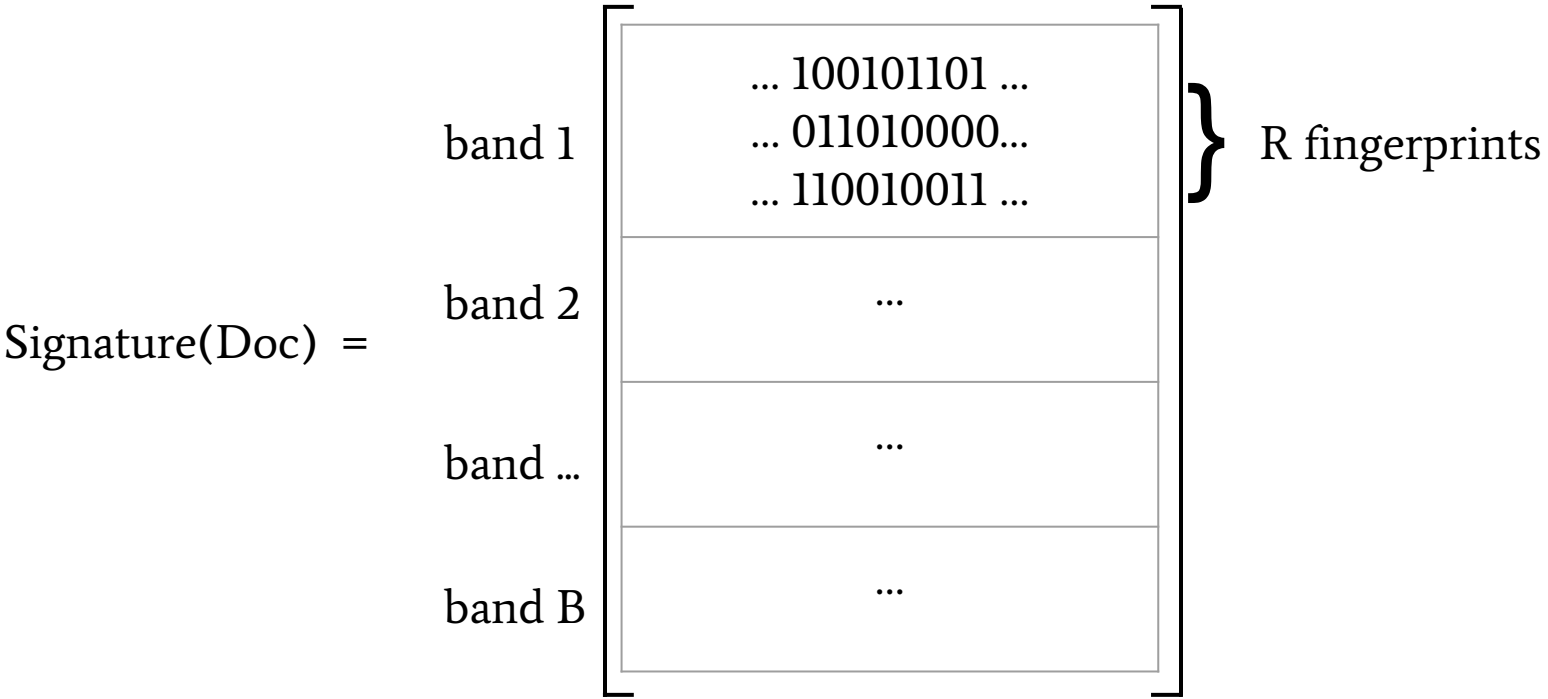
MinHash - Implementation

1. Generate the fingerprints of the document
2. Define K hash functions: $h_1, h_2, \ldots, h_k$.
3. Define $Sig(Doc) = [h_1(Doc), h_2(Doc), \ldots, h_k(Doc)]$
4. Define $O = \{ i / h_i(Doc_1) = h_i(Doc_2) \}$
5. $\mathbf{Sim(Doc_1, Doc_2) = \frac{|O|}{K} \simeq Jaccard(Doc_1, Doc_2)}$

Storage $\cong$	4 byte * **K** * #Docs	With **K** << Doc_length
Complexity $\cong$	**K** * #Docs	

Local Sensitive Hashing

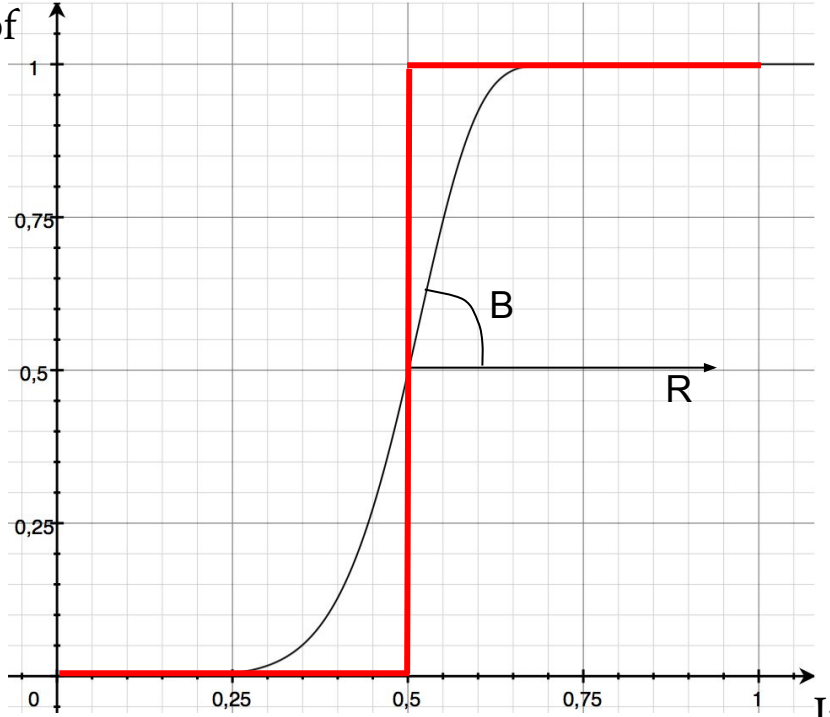
Divide the signature Sig(Doc) into **B bands of R rows each**, such that B*R = K:



Local Sensitive Hashing - Analysis

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

Probability of becoming a candidate



Jaccard of documents

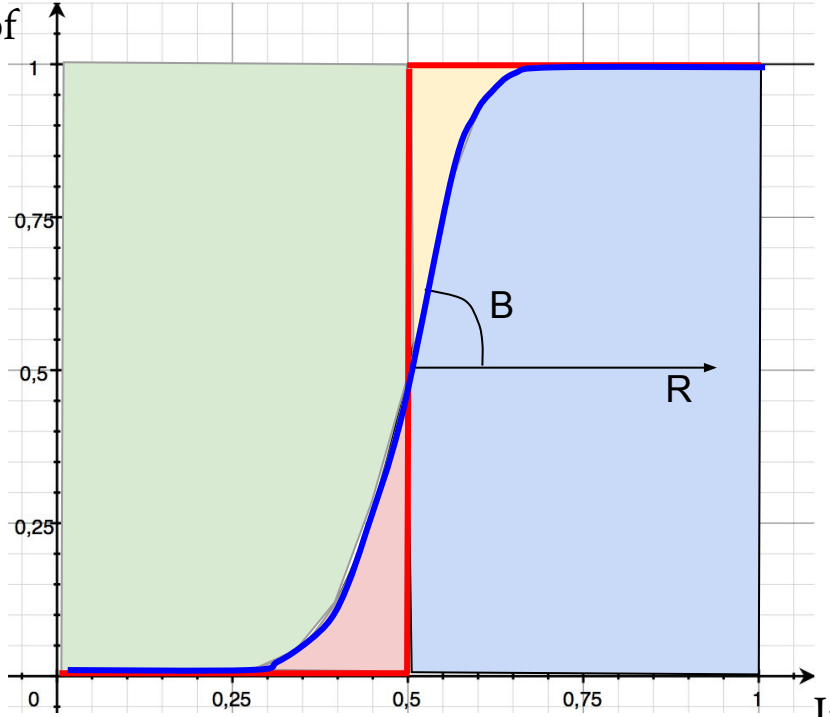
S-curve

- **Threshold** ≅ (1/B)^(1/R)

Local Sensitive Hashing - Analysis

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

Probability of becoming a candidate



Jaccard of documents

S-curve

- **Threshold** ≅ (1/B)^(1/R)
- True Positive
- True Negative
- False Positive
- False Negative

Probabilistic search - Results

From:

Storage	≅	Shingle_length * Doc_length * #Docs
Complexity	≅	Doc_length * #Docs

To:

Storage	≅	4 byte * K * #Docs
Complexity	≅	K * #Docs * p(“candidate”)

With K << Doc_length and p(“candidate”) << 1

Probabilistic search - Results

Naïve approach		Min hash + LSH
2 B files	VS	**2 B** files
1 PB of index		**1,5 TB** of index
Slow search		Fast search & update

Thank you

...

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

•••

Any questions?