Bloom Filters

Credits: M. Mitzenmacher

Bloom Filters: High Level Idea

- Everyone thinks they need to know exactly what everyone else has. *Give me a list of what you have.*
- Lists are long and unwieldy.
- Using Bloom filters, you can get small, approximate lists. *Give me information so I can figure out what you have.*

Lookup Problem

- Given a set $S = \{x_1, x_2, x_3, \ldots x_n\}$ on a universe U, want to answer queries of the form:

$$Is\ y \in S.$$

- Example: a set of URLs from the universe of all possible URL strings.
- Bloom filter provides an answer in
  - "Constant" time (time to hash).
  - Small amount of space.
  - But with some probability of being wrong.

Bloom Filters

Start with an m bit array, filled with 0s.

B

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hash each item x_j in S k times. If $H_i(x_j) = a$, set $B[a] = 1$.

B

0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

To check if y is in S, check B at $H_i(y)$. All k values must be 1.

B

0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

Possible to have a false positive; all k values are 1, but y is not in S.

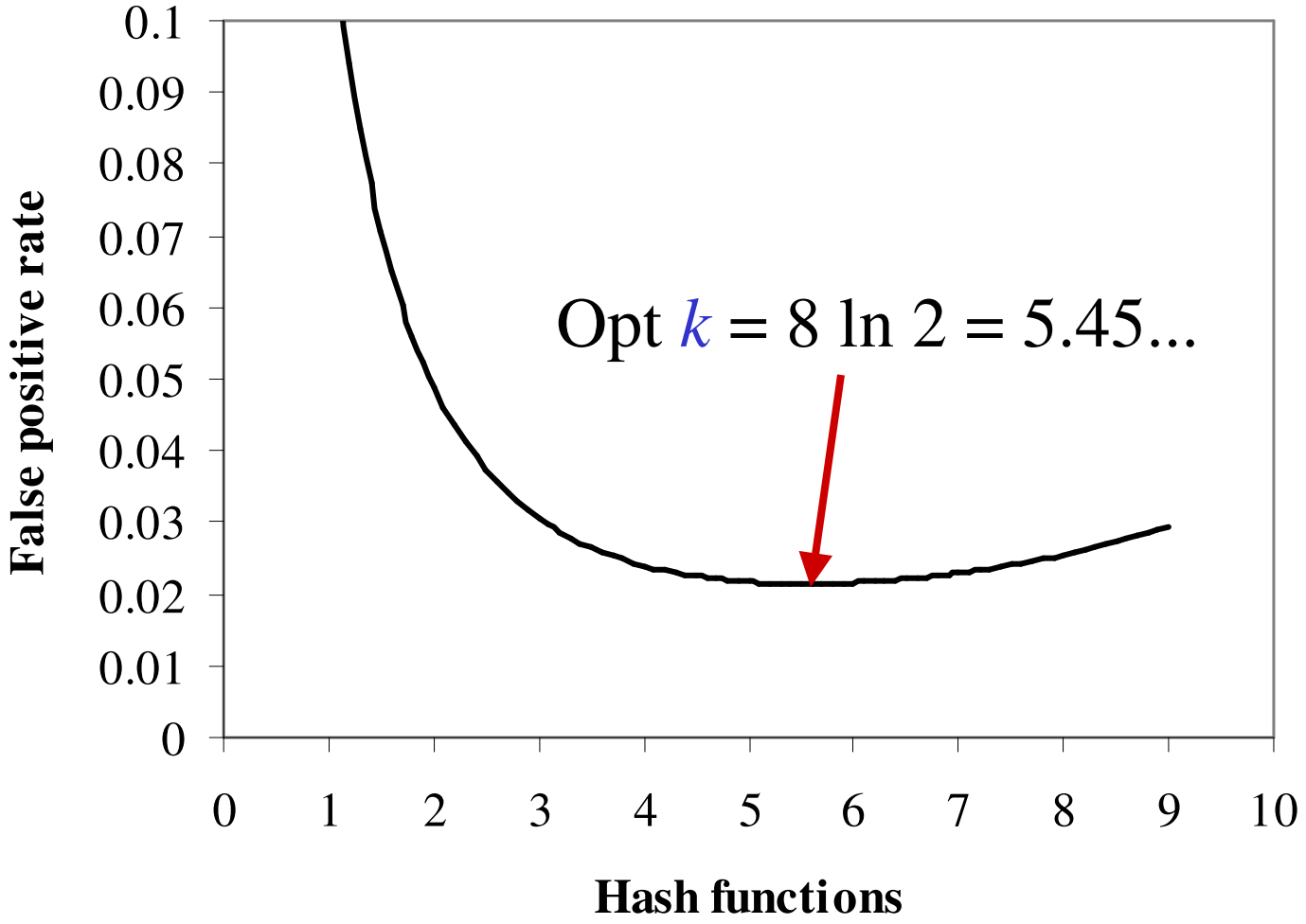
B

0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

Errors

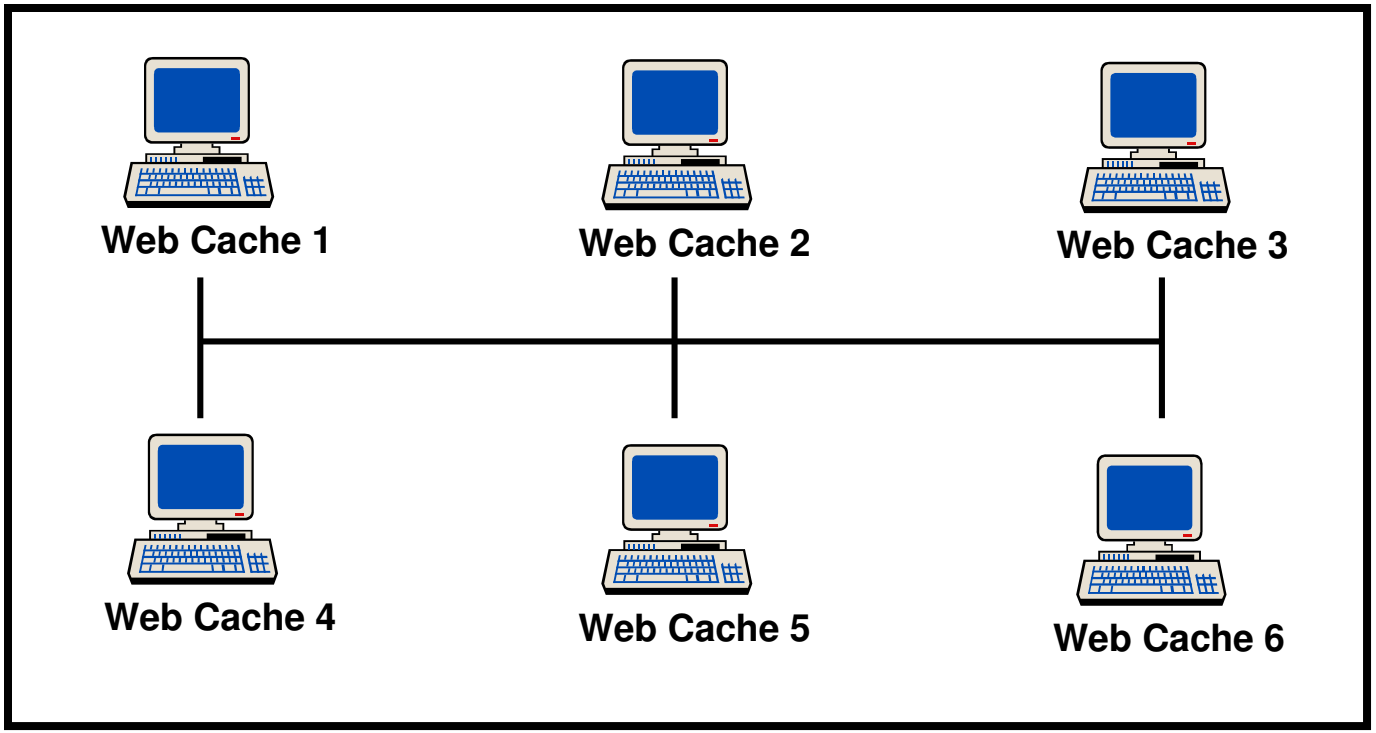
- Assumption: We have good hash functions, look random.
- Given m bits for filter and n elements, choose number k of hash functions to minimize false positives:
  - Let $p = \Pr[\text{cell is empty}] = (1 - 1/m)^{kn} \approx e^{-kn/m}$
  - Let $f = \Pr[\text{false pos}] = (1 - p)^k \approx (1 - e^{-kn/m})^k$
- As k increases, more chances to find a 0, but more 1's in the array.
- Find optimal at $k = (\ln 2)m/n$ by calculus.

Example


0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
False positive rate
Opt k = 8 ln 2 = 5.45...
0
1
2
3
4
5
6
7
8
9
10
Hash functions

$m/n = 8$

Bloom Filters: Distributed Systems



- Send Bloom filters of URLs.
- False positives do not hurt much.
  - Get errors from cache changes anyway.

Tradeoffs

- Three parameters.
  - Size m/n : bits per item.
  - Time k : number of hash functions.
  - Error f : false positive probability.

Compression

- Insight: Bloom filter is not just a data structure, it is also a message.
- If the Bloom filter is a message, worthwhile to compress it.
- Compressing bit vectors is easy.
  - Arithmetic coding gets close to entropy.
- Can Bloom filters be compressed?

Optimization, then Compression

- Optimize to minimize false positive.

$$p = \Pr[\text{cell is empty}] = (1 - 1/m)^{kn} \approx e^{-kn/m}$$

$$f = \Pr[\text{false pos}] = (1 - p)^k \approx (1 - e^{-kn/m})^k$$

$$k = (m \ln 2)/n \text{ is optimal}$$

- At $k = m\ (\ln 2)\ /n$, $p = 1/2$.
- Bloom filter looks like a random string.
  - Can't compress it.

Tradeoffs

- With compression, four parameters.
  - Compressed (transmission) size z/n : bits per item.
  - Decompressed (stored) size m/n : bits per item.
  - Time k : number of hash functions.
  - Error f : false positive probability.

Does Compression Help?

- Claim: transmission cost limiting factor.
  - Updates happen frequently.
  - Machine memory is cheap.
- Can we reduce false positive rate by
  - Increasing decompressed size (storage).
  - Keeping transmission cost constant.

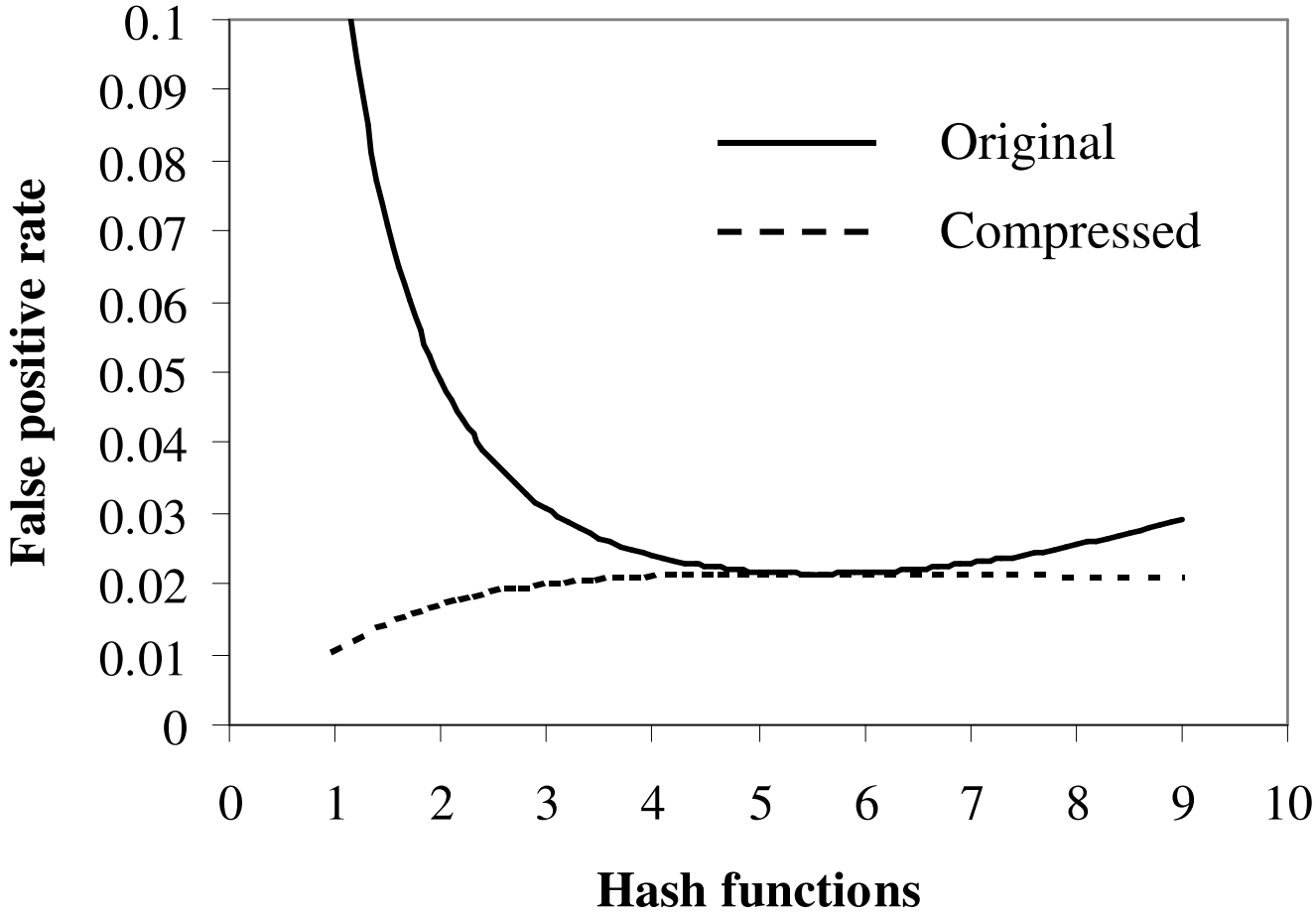
Errors: Compressed Filter

- Assumption: optimal compressor, $z = m\mathrm{H}(p)$.
  - $\mathrm{H}(p)$ is entropy function; optimally get $\mathrm{H}(p)$ compressed bits per original table bit.
  - Arithmetic coding close to optimal.
- Optimization: Given z bits for compressed filter and n elements, choose table size m and number of hash functions k to minimize f.

$$p \approx e^{-kn/m}; f \approx (1-e^{-kn/m})^k; z \approx mH(p)$$

- Optimal found by calculus.

Example


0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
False positive rate
Original
Compressed
0
1
2
3
4
5
6
7
8
9
10
Hash functions

$z/n = 8$

Results

- At $k = m\ (\ln 2)\ /n$, false positives are maximized with a compressed Bloom filter.
  - Best case without compression is worst case with compression; compression always helps.
- Side benefit: Use fewer hash functions with compression; possible speedup.

Examples

Array bits per elt.	m/n	8	14	92
Trans. Bits per elt.	z/n	8	7.923	7.923
Hash functions	k	6	2	1
False positive rate	f	0.0216	0.0177	0.0108
Array bits per elt.	m/n	16	28	48
Trans. Bits per elt.	z/n	16	15.846	15.829
Hash functions	k	11	4	3
False positive rate	f	4.59E-04	3.14E-04	2.22E-04

- Examples for bounded transmission size.
  - 20-50% of false positive rate.
- Simulations very close.
  - Small overhead, variation in compression.

Examples

Array bits per elt.	m/n	8	12.6	46
Trans. Bits per elt.	z/n	8	7.582	6.891
Hash functions	k	6	2	1
False positive rate	f	0.0216	0.0216	0.0215
Array bits per elt.	m/n	16	37.5	93
Trans. Bits per elt.	z/n	16	14.666	13.815
Hash functions	k	11	3	2
False positive rate	f	4.59E-04	4.54E-04	4.53E-04

- Examples with fixed false probability rate.
  - 5-15% compression for transmission size.
- Matches simulations.

Bloom Filters: Other Applications?

- Put HERE your ideas

Conclusions

- There are lots of interesting problems out there.
  - New techniques, algorithms, data structures
  - New analyses
  - Finding the right way to apply known ideas