A Little Confusing

"Without [a block digest], one must query the offset digest with all possible offsets: although the extra space afforded by not having a block digest increases the accuracy of the offset digest, the testing of every offset gives both designs roughly **equivalent** accuracy...So, we can omit the block digest and save storage to **increase the accuracy** of the offset digest".

Compressed Bloom Filters

"Compressed Bloom Filters", M. Mitzenmacher, *IEEE/ACM Transactions on Networking*, 2002.

Inspired by "Summary Cache" paper.

Compress a Bloom Filter... Wha?

Recall that a Bloom Filter optimized for minimum false positive rate is half ones, half zeros.

m-bit buffer, $\frac{m}{2}$ randomly distributed ones, the rest zeros. Seems pretty uncompressable, right?

Wrong Approach to Compression

Right. But that's not exactly what we want to compress...

Traditional constraints of BF: memory available (m), number of elements (n), minimizing false positives by manipulating k with respect to m and n, using k that parallelizes well.

Summary Cache introduced new constraint: **Transmission size** (z).

Compression Assumptions

Can store larger filter in memory than we are willing to transmit

Willing to take a processing hit for compression and decompression upon transmitting and receiving

Tweak k or m to achieve better compression with low False Positives.

- Reduce FP for desired compressed size
- Compress to smaller transmission size for fixed FP

Using Eight Bits per Element Compressed

Array bits per element	m/n	8	14	92
Transmission bits per element	z/n	8	7.923	7.923
No. Hash Functions	k	6	2	1
False Positive Rate	f	.0216	.0177	.0108

Maintaining FP Rate around .02

Array bits per element	m/n	8	12.6	46
Transmission bits per element	z/n	8	7.582	6.891
No. Hash Functions	k	6	2	1
False Positive Rate	f	.0216	.0216	.0215

Other Uses of Compression with Summary Cache

Compressing deltas of filters

Compressing counting filters (not actually transmitted in Summary Cache)

Misleading Mention of Compression

From Section 5.2 "Resource Requirements" of HBF paper

"By populating the bit-vector sparsely (by choosing $k \ll m/n$), however, it would [sic] possible to compress the Bloom filters better and at the same time improve the false positive rates [21] at the cost of more high-speed memory at the network component".

Recall that compressed filters are not intended to reduce **storage** size but **transmission** size.

From referenced paper: "We assume here that all lookup computation on the Bloom filter is done after decompression at the proxies...achieving random access, efficiency, and good compression simultaneously in generally difficult..."