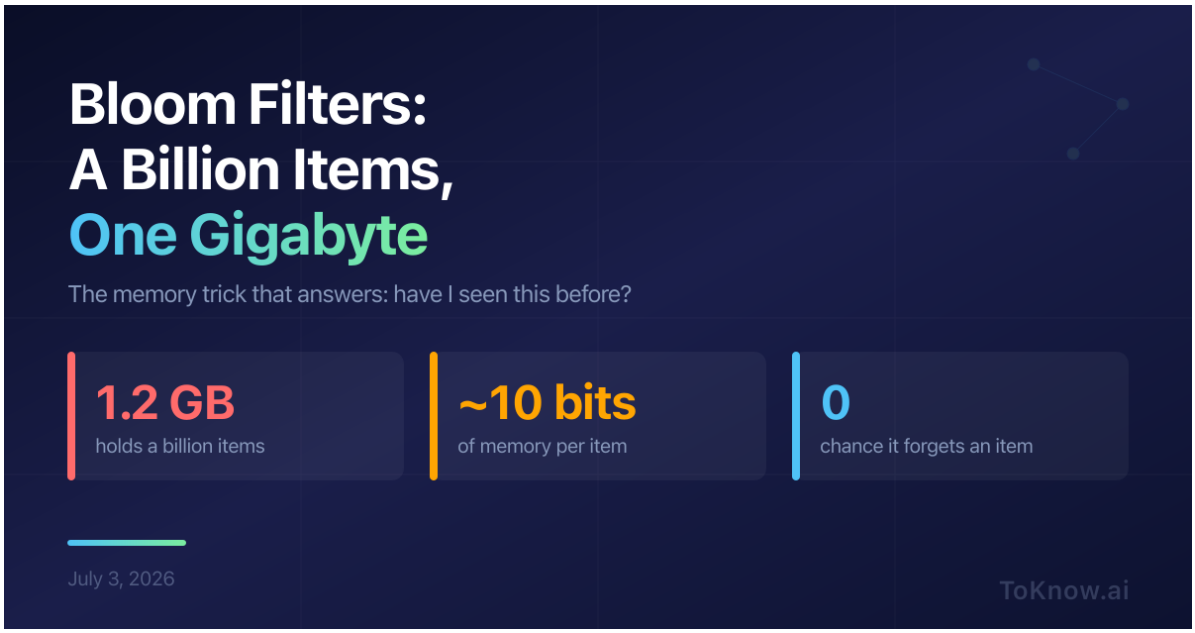
Bloom Filters: How Apps Remember a Billion Things in About a Gigabyte

Kabui, Charles

2026-07-03

Read at ToKnow.ai



A Bloom filter is a fast way to answer one question: have I seen this before? Picture a row of light switches, all off. To record an item, you run it through a few fixed hash functions, formulas that turn any input into switch positions, and flip those switches on. To check the item later, you look at those same switches. If even one is off, the item is definitely new. If

all are on, you have probably seen it, though not for sure, since other items may have flipped the same switches. It never stores the items, only the bits.

That gives two answers: a definite no, or a probably yes, and the trade pays off at scale. Take a dating app with 50 million users: on every swipe it asks whether these two have already been shown to each other. Checking a database that often would overwhelm it, and an exact list of every pair would eat memory. A Bloom filter answers instantly. It never shows a duplicate, since a definite no is always right, and only occasionally hides a fresh profile on a wrong probably yes. A billion pairs at a 1% error rate fit in about 1.2 GB, over ten times smaller than an exact set.

Accepting a few false alarms to save that much memory is a trade real systems make constantly. Databases like Cassandra and Google's Bigtable use Bloom filters to skip pointless disk reads. Akamai uses them to avoid caching pages viewed only once, nearly three-quarters of what it sees. Web crawlers use them to track which links they have already visited. One small idea, running software used by billions.

Sources:

- Jason Davies: Bloom Filters (interactive demo)
- Wikipedia: Bloom filter
- Apache Cassandra: Bloom Filters
- Maggs & Sitaraman: Algorithmic Nuggets in Content Delivery (PDF)
- Burton H. Bloom (1970): Space/Time Trade-offs in Hash Coding with Allowable Errors (PDF)
