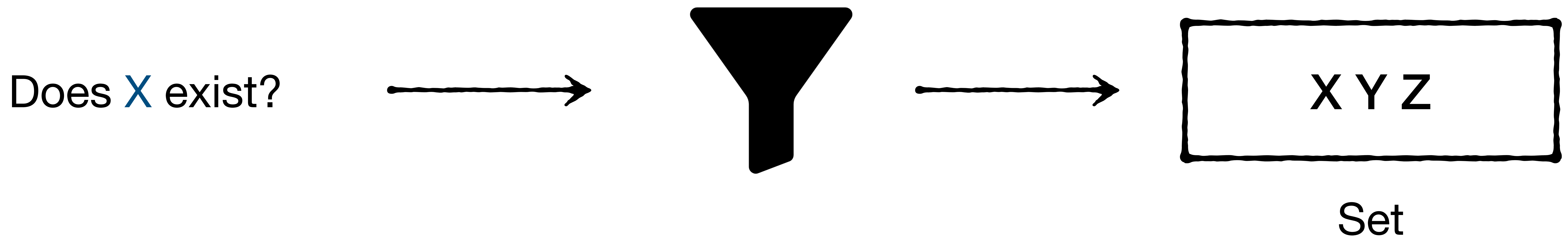
Adaptive Filters

How to learn from your mistakes

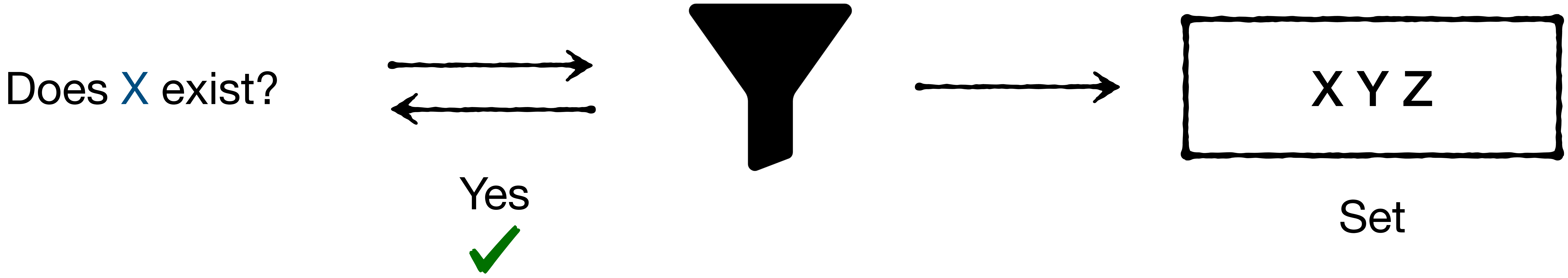
Prashant Pandey, University of Utah
https://prashantpandey.github.io/

What is a filter data structure?



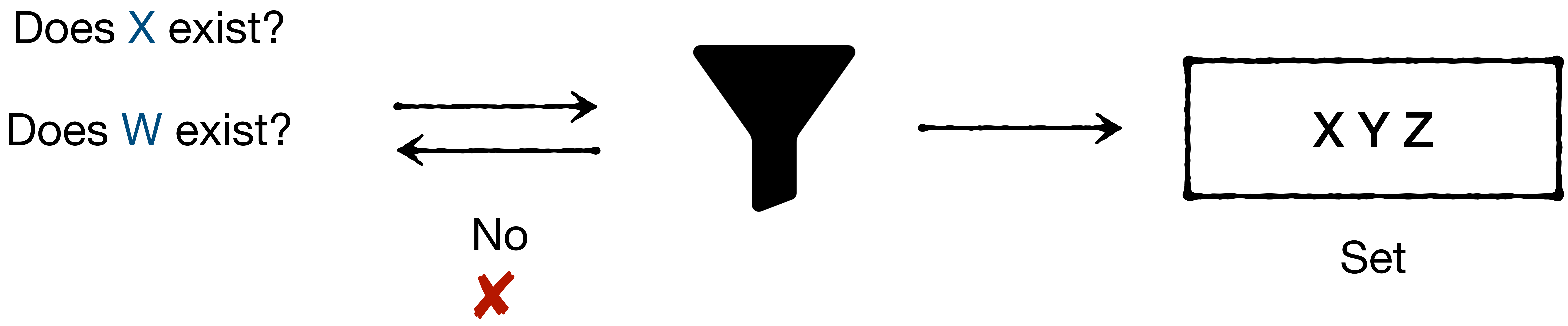
A filter compactly represents a set by trading off accuracy for space efficiency

What is a filter data structure?



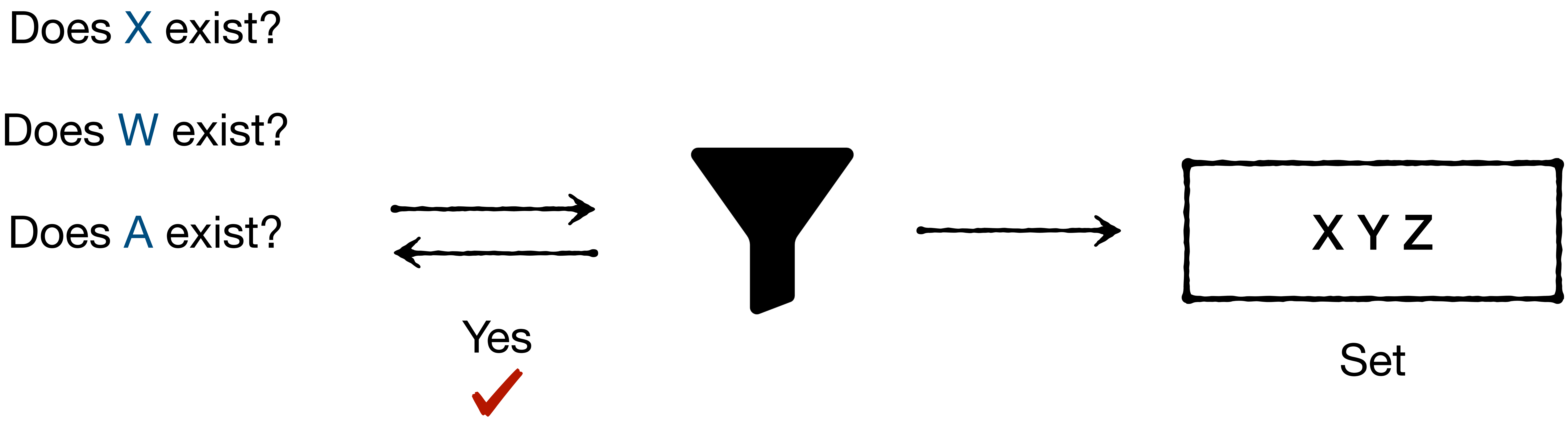
A filter compactly represents a set by trading off accuracy for space efficiency

What is a filter data structure?



A filter compactly represents a set by trading off accuracy for space efficiency

What is a filter data structure?



A filter compactly represents a set by trading off accuracy for space efficiency

A filter guarantees a false-positive rate ϵ

$q =$ **query item** $S =$ **set of items**

if q ∈ S, return	**True** with probability 1	true positive
if q ∉ S, return	**False** with probability $> 1 - \epsilon$	true negative
	True with probability $\leq \epsilon$	false positive

One-sided errors

False positives with tunable probability

False-positives enable filters to be compact

$n =$ **number of items** $U =$ **universe of items**

space $\geq n \log(1/\epsilon)$

Small

space $= \Omega(n \log(|U|))$

Large

Filter

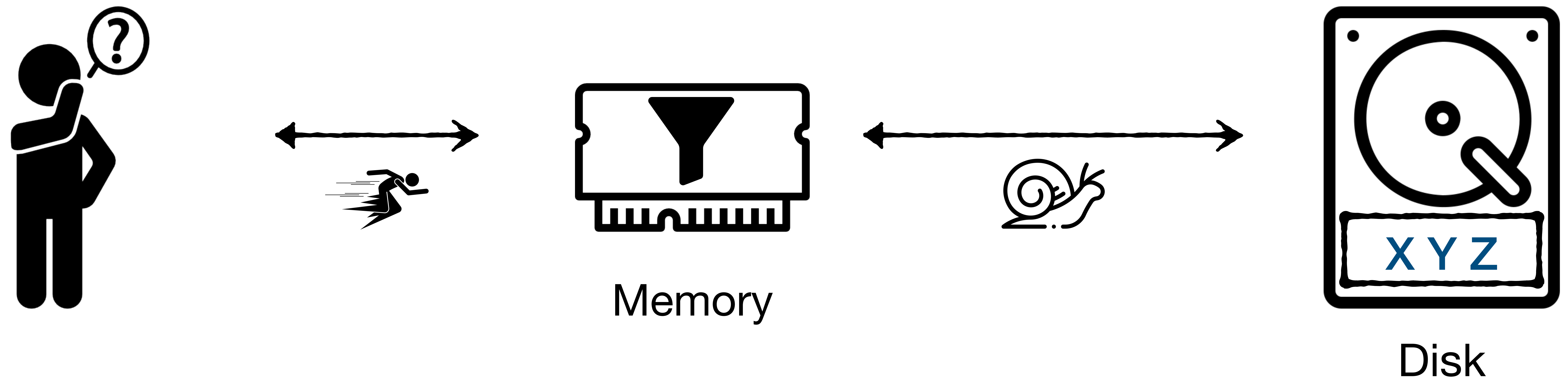
Hash table/Tree

For $\epsilon = 2\,\%$, filters require ~1 Byte/item. Hash table/Tree can take >8-16 Byte/item.

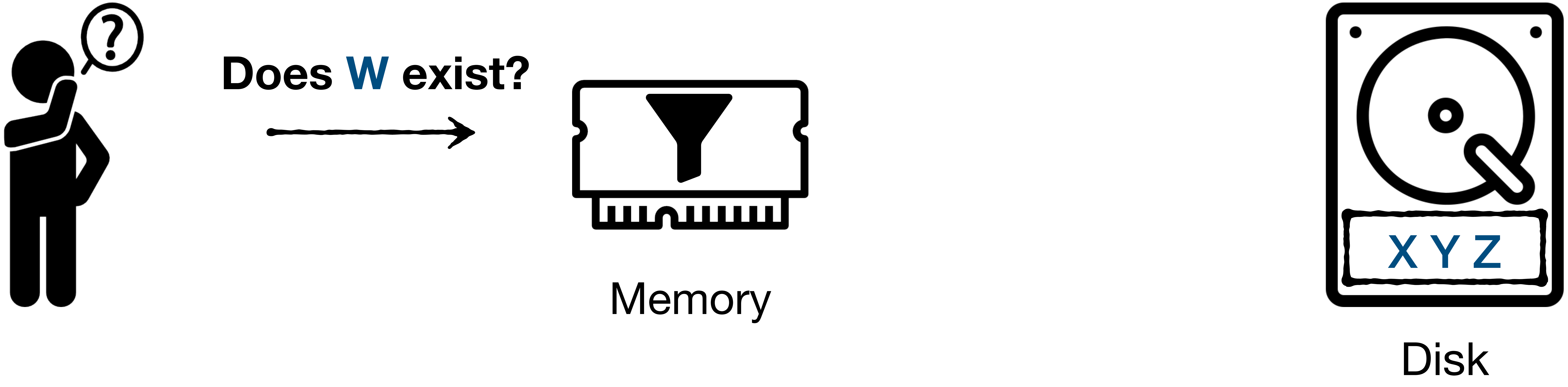
Filters offer weak guarantees

The maximum false positive rate is only guaranteed for a single query and not an arbitrary sequence of queries

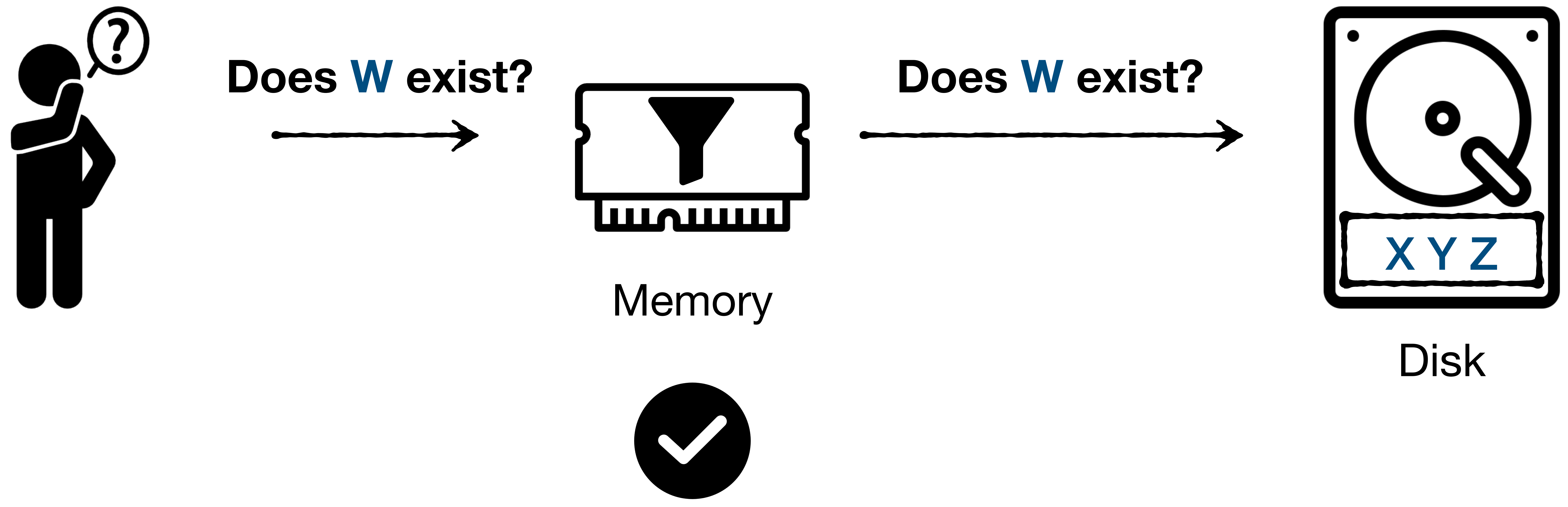
Skewed workloads can make filters obsolete



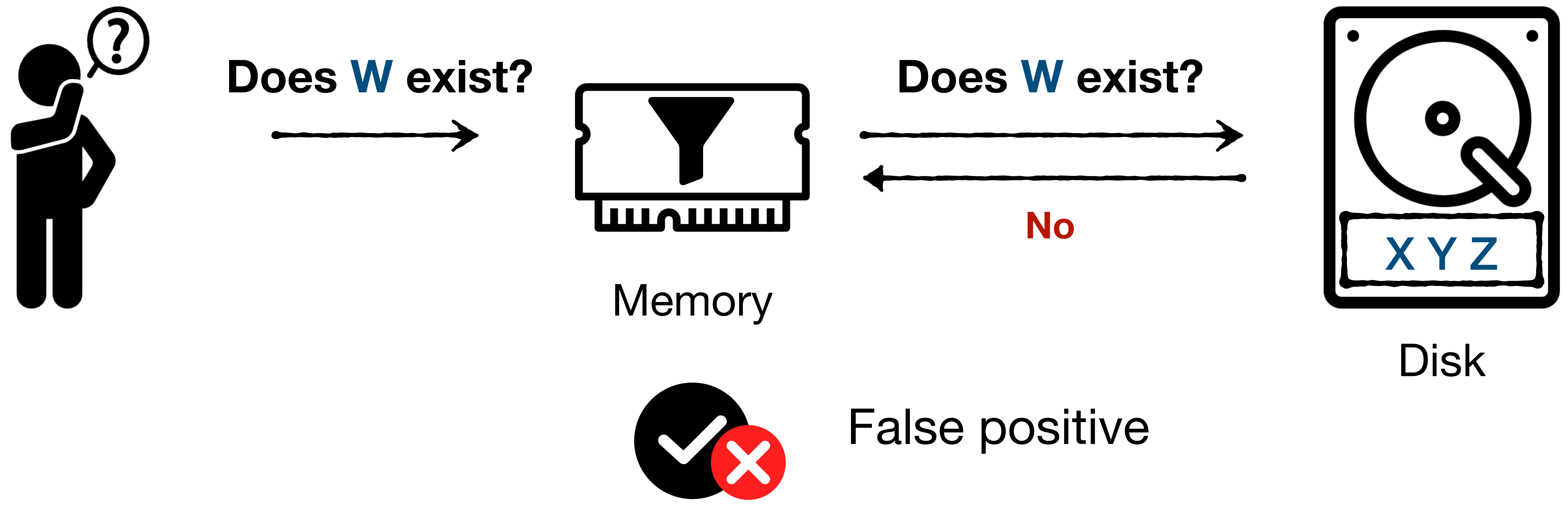
Skewed workloads can make filters obsolete



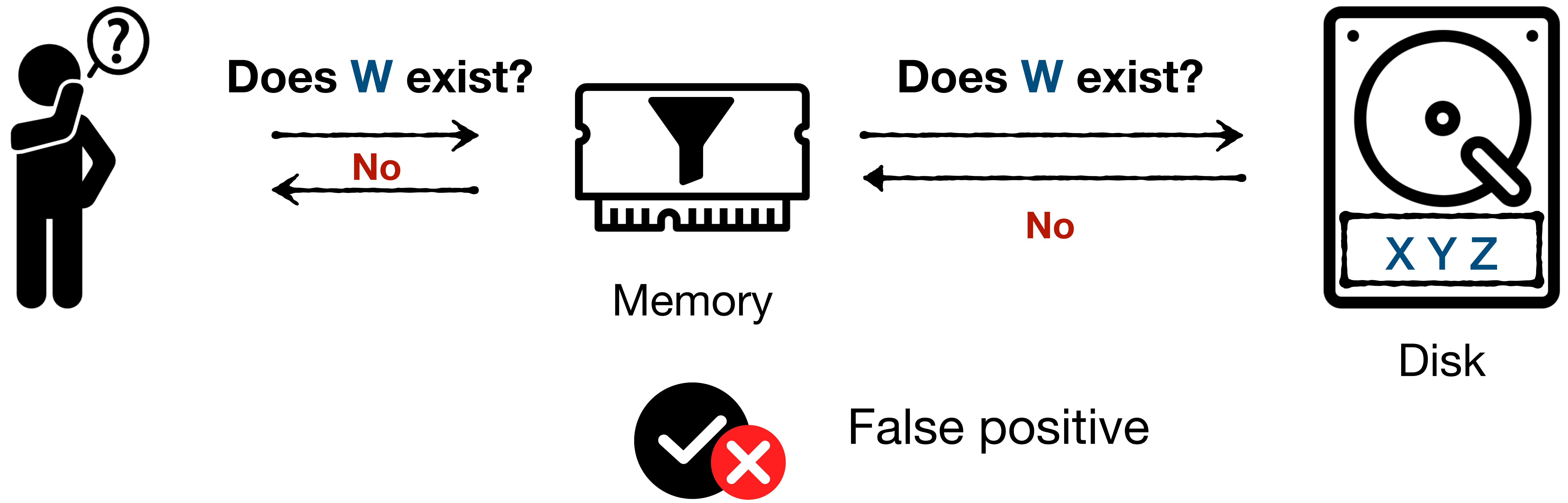
Skewed workloads can make filters obsolete



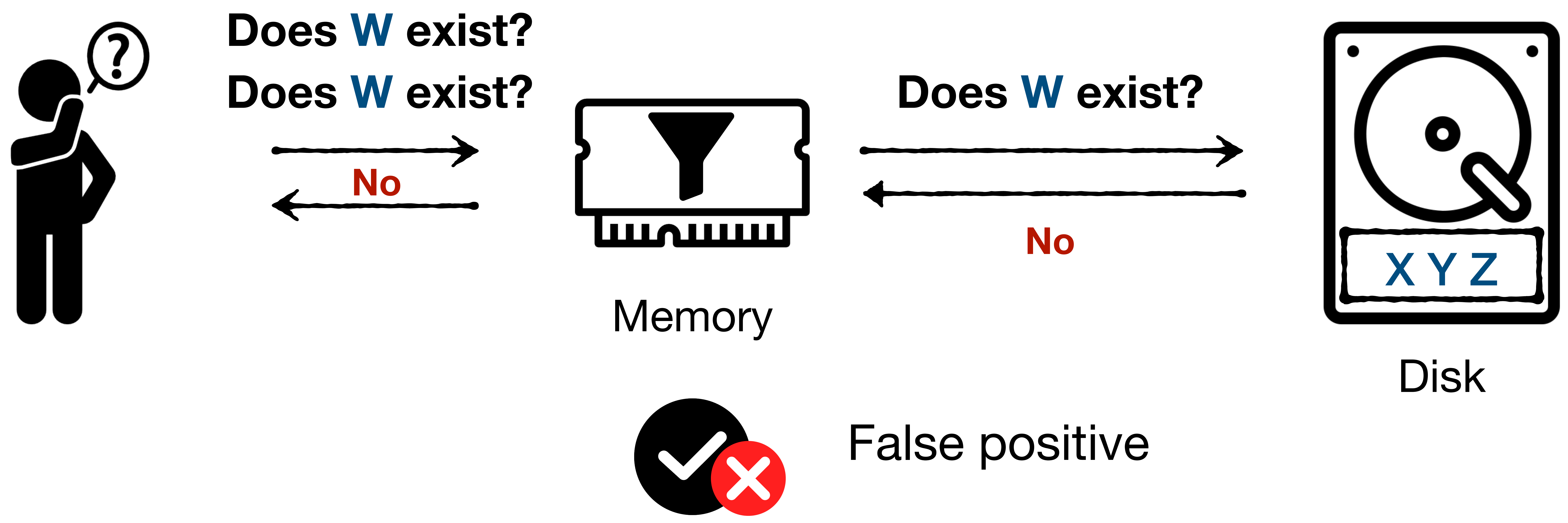
Skewed workloads can make filters obsolete



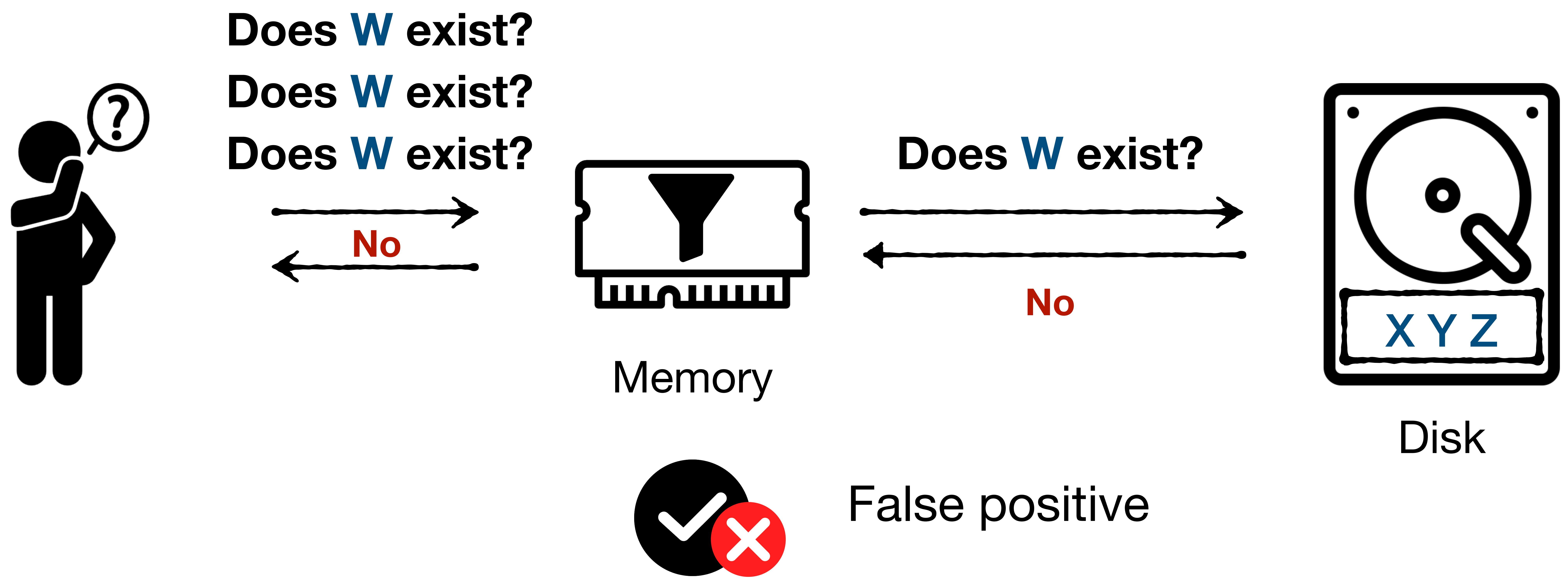
Skewed workloads can make filters obsolete



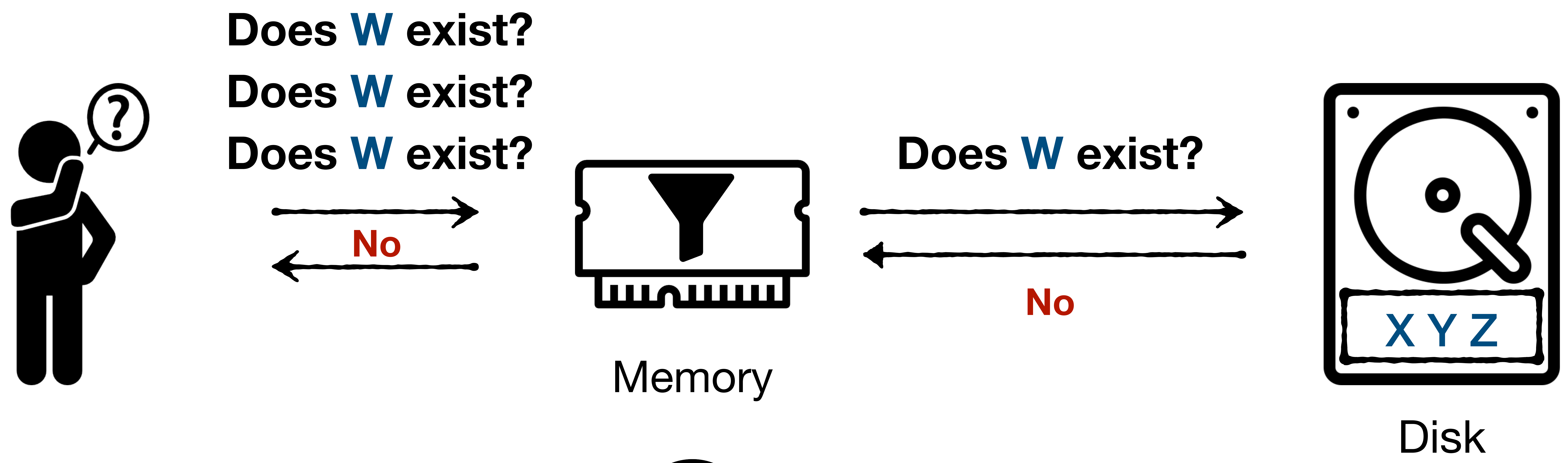
Skewed workloads can make filters obsolete

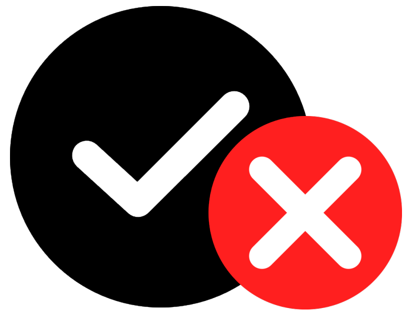


Skewed workloads can make filters obsolete



Skewed workloads can make filters obsolete



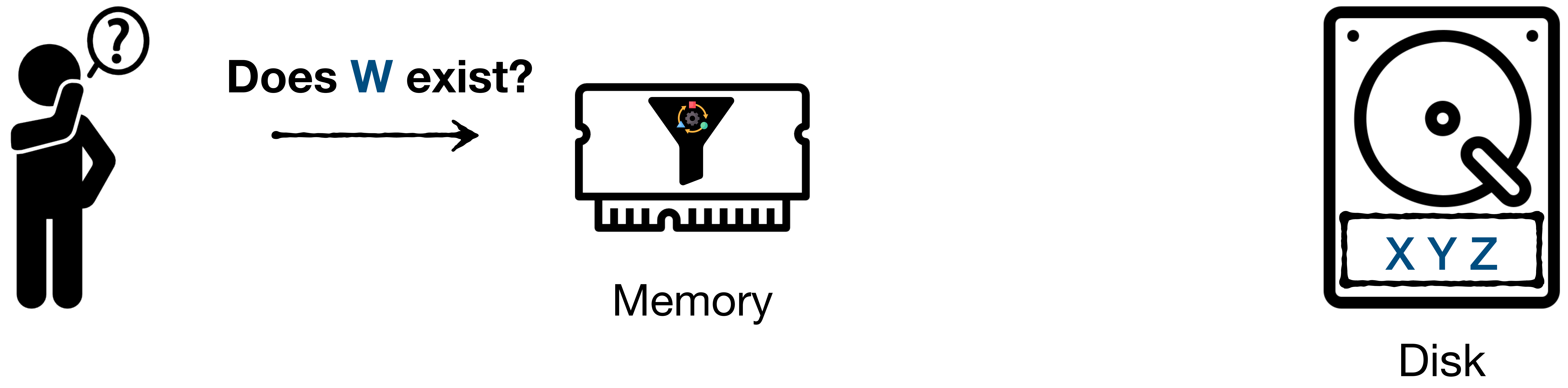


False positive

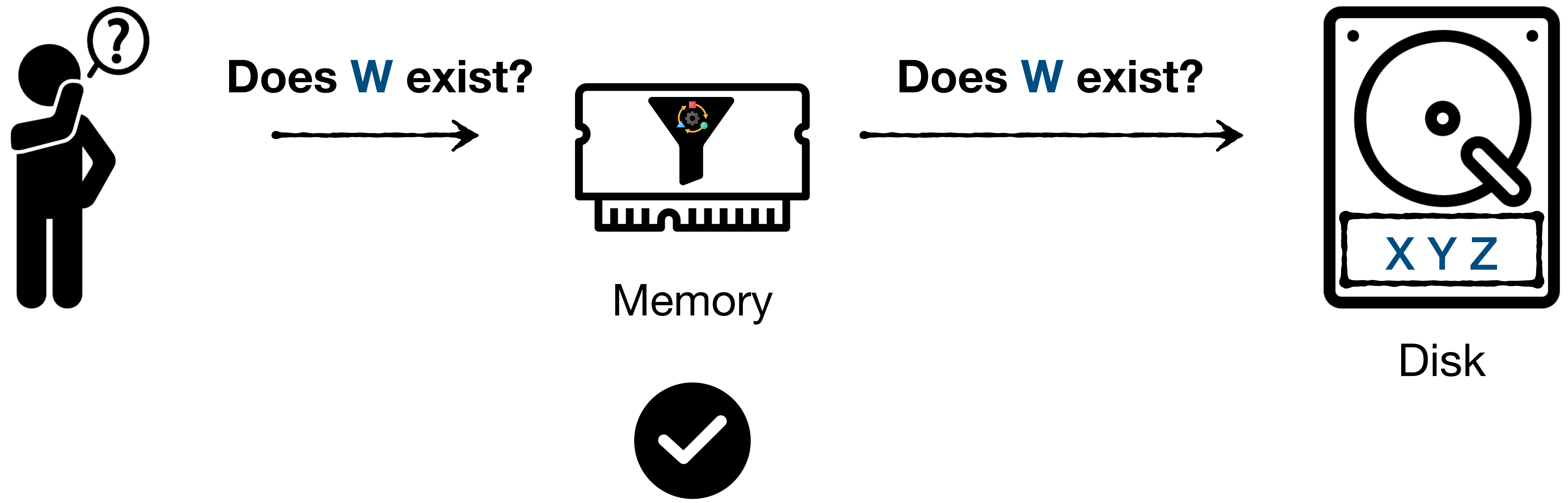
False-positive rate $\leq \epsilon$, only for a single query

Can we learn from the feedback?

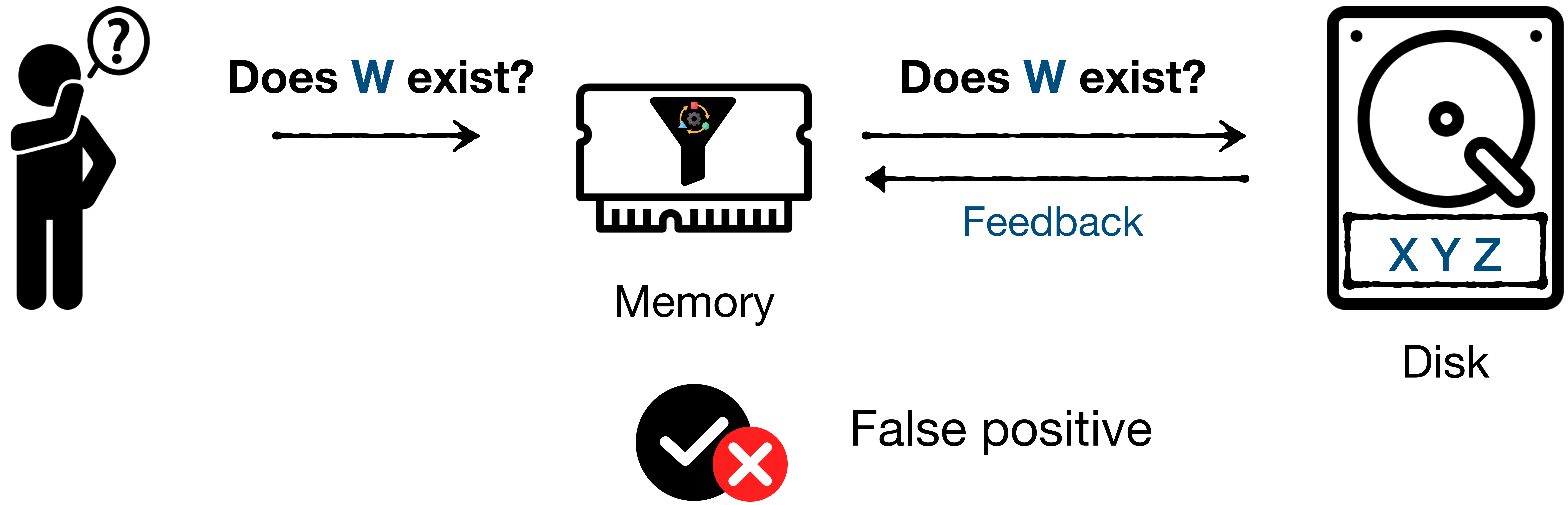
Adaptive filters change their state upon feedback



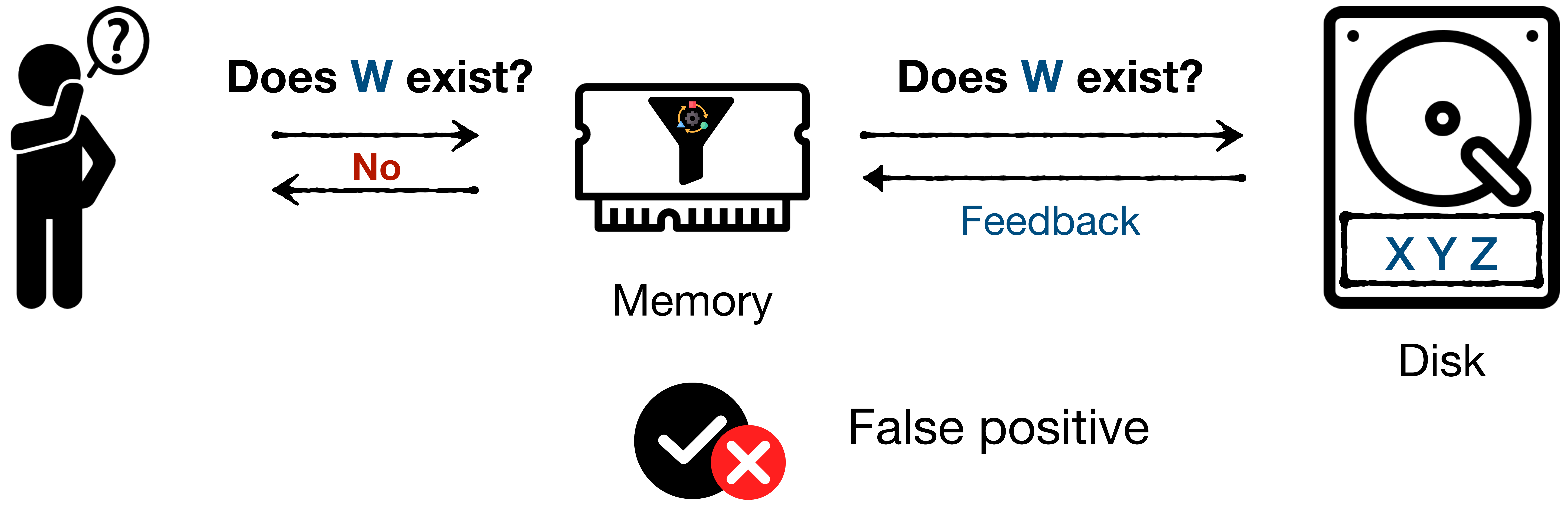
Adaptive filters change their state upon feedback



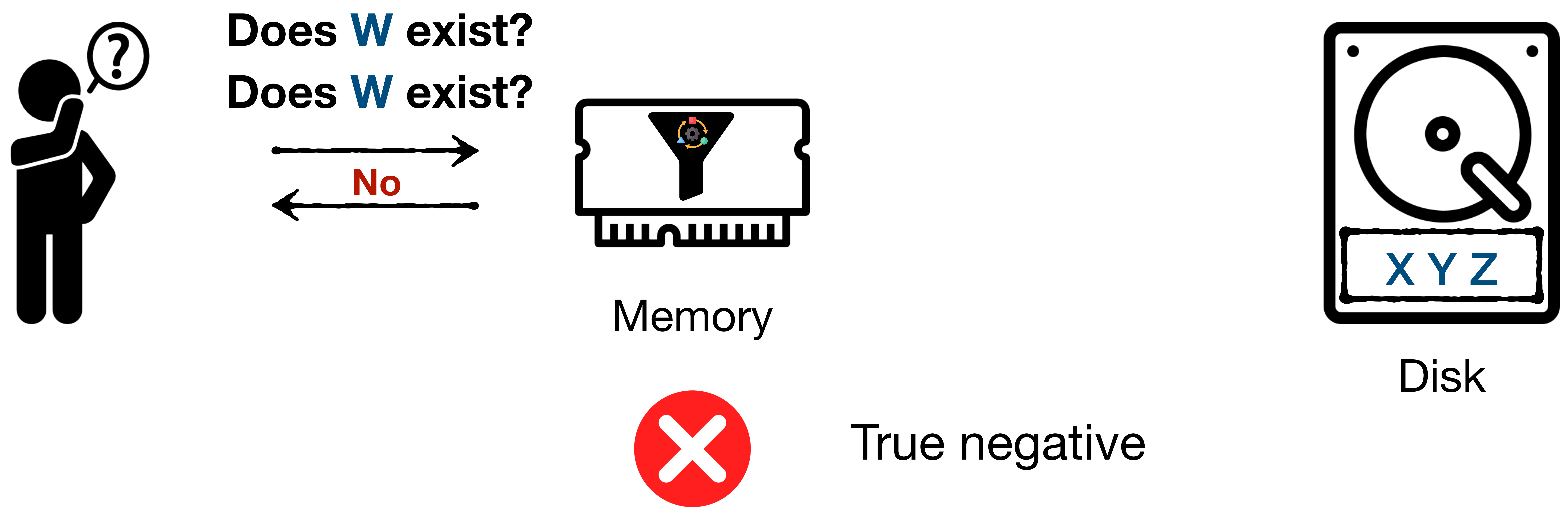
Adaptive filters change their state upon feedback



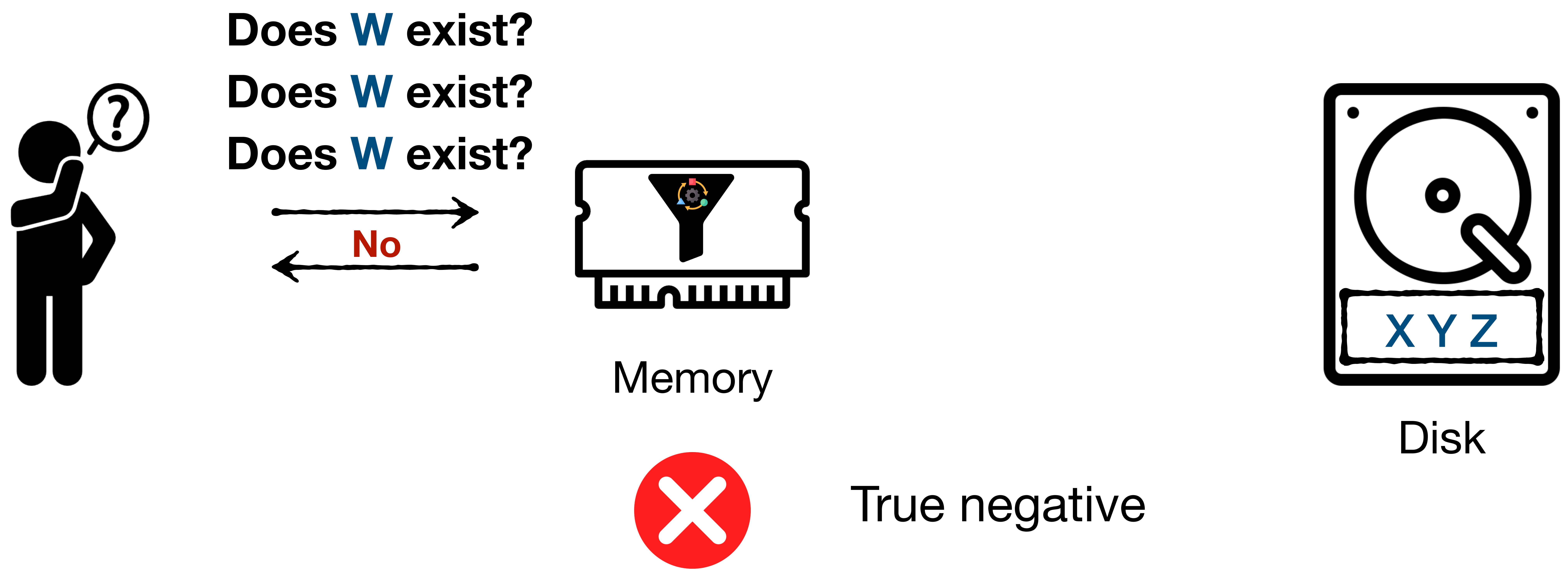
Adaptive filters change their state upon feedback



Adaptive filters change their state upon feedback



Adaptive filters change their state upon feedback

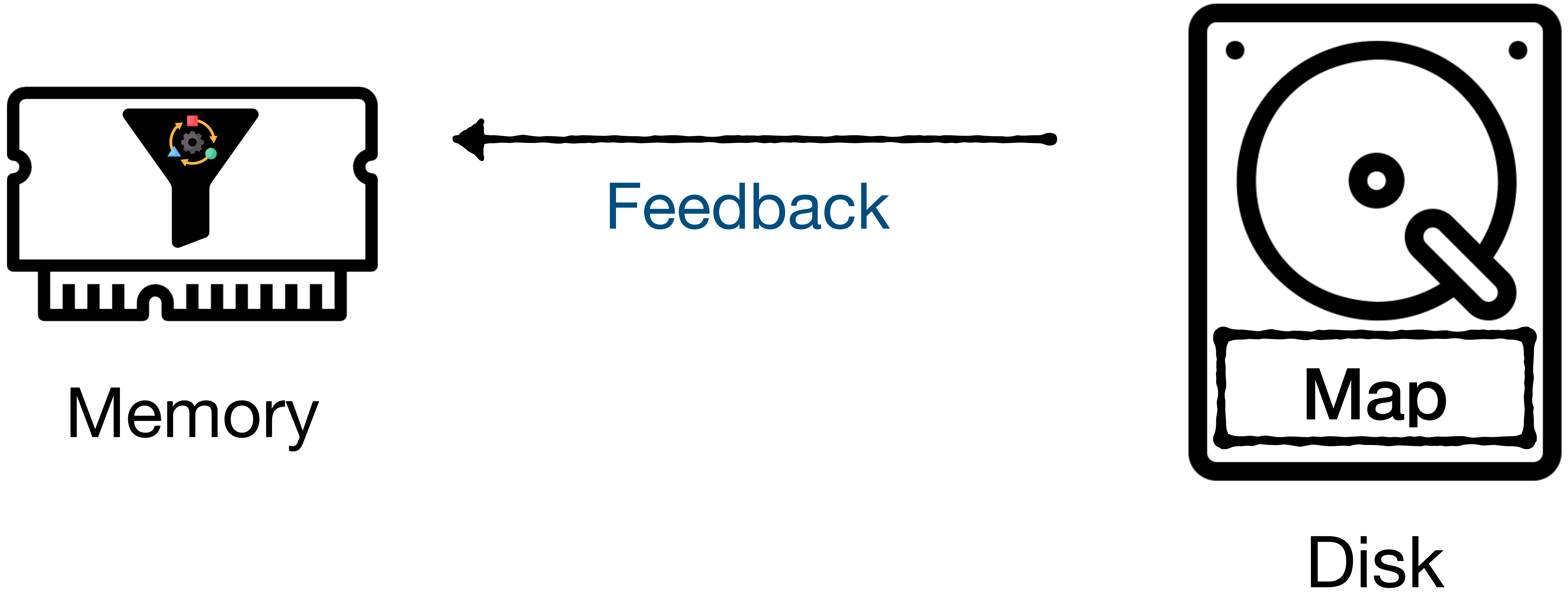


Adaptive filters [BFG+ 2018]

An adaptive filter modifies its state upon feedback and produces close to $O(\epsilon n)$ false positives for any sequence of n queries

False-positive rate $\leq \epsilon$, independent of the query distribution

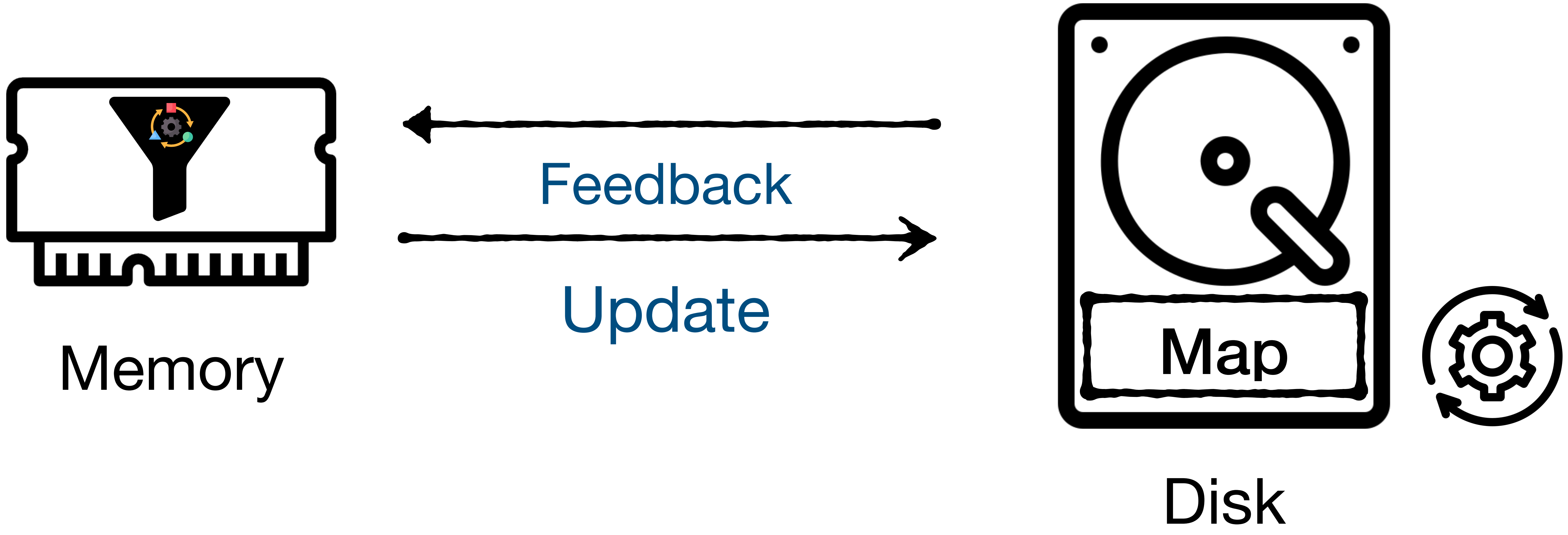
Adaptive filter design has two parts [BFG+ 2018]



Small in-memory filter
accessed on every query

Large disk-resident map
accessed during adaptations

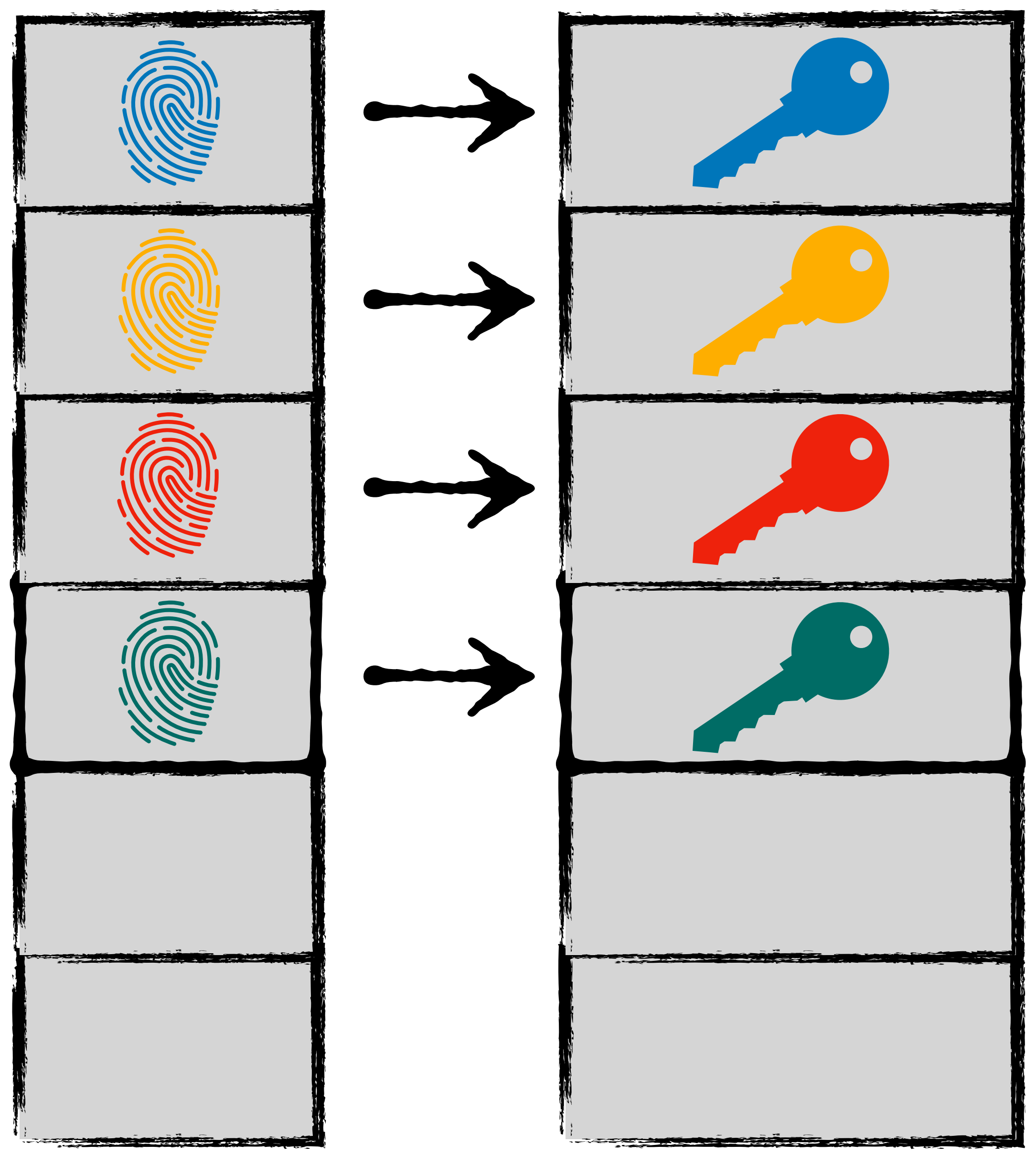
Adaptive filter design has two parts [BFG+ 2018]



On-disk map enables adaptations and is updated to fix fingerprint collisions

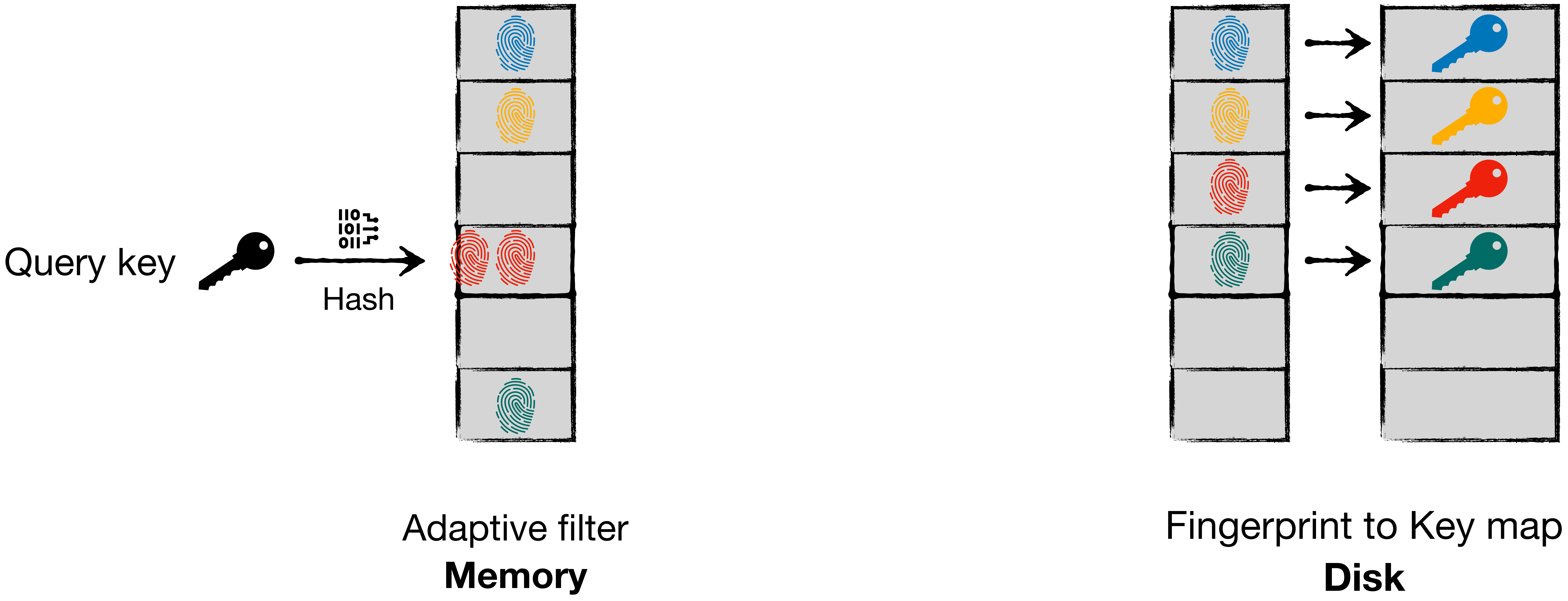
Adaptive filters employ variable-length fingerprints

Adaptive filter
Memory



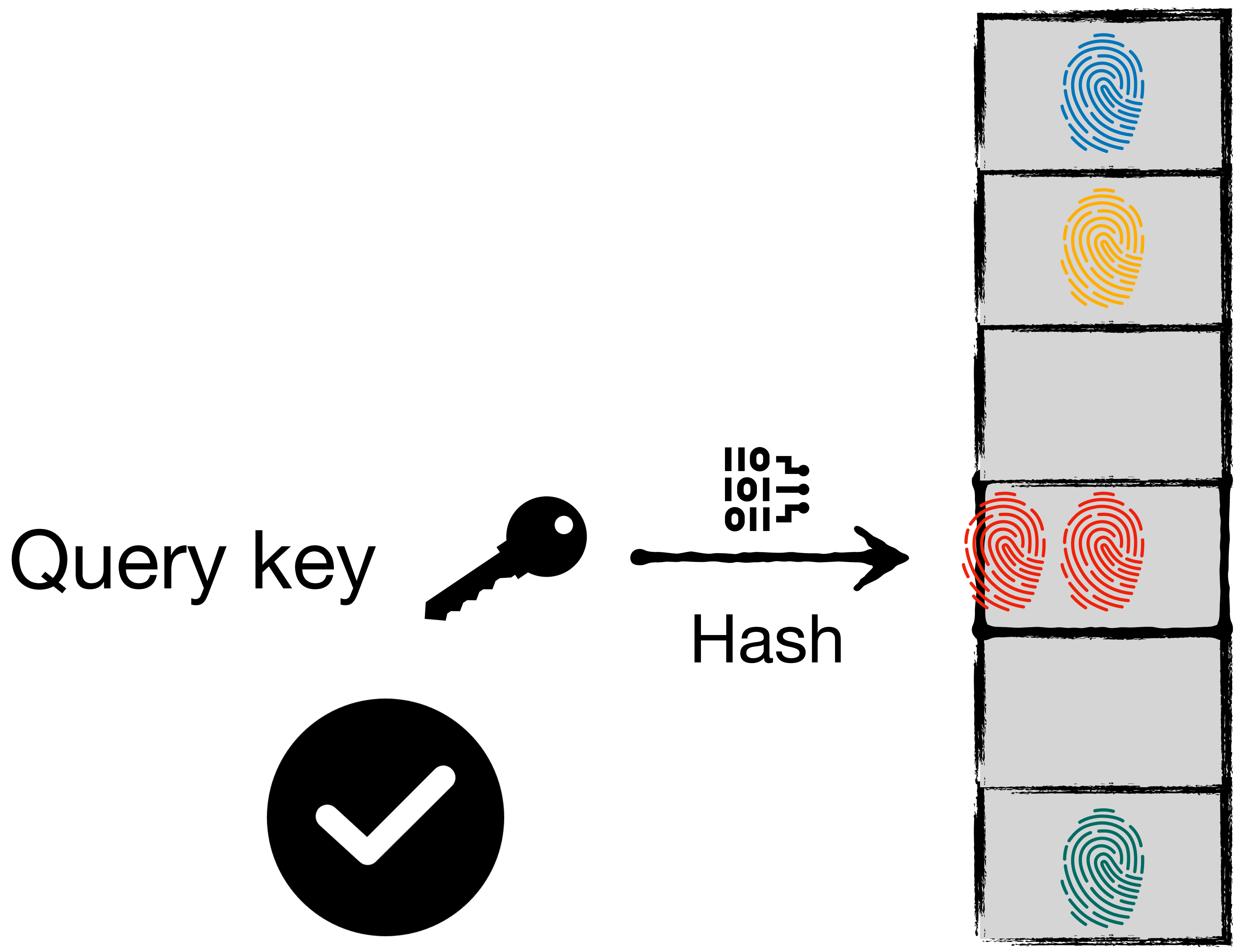
Fingerprint to Key map
Disk

Adaptive filters employ variable-length fingerprints

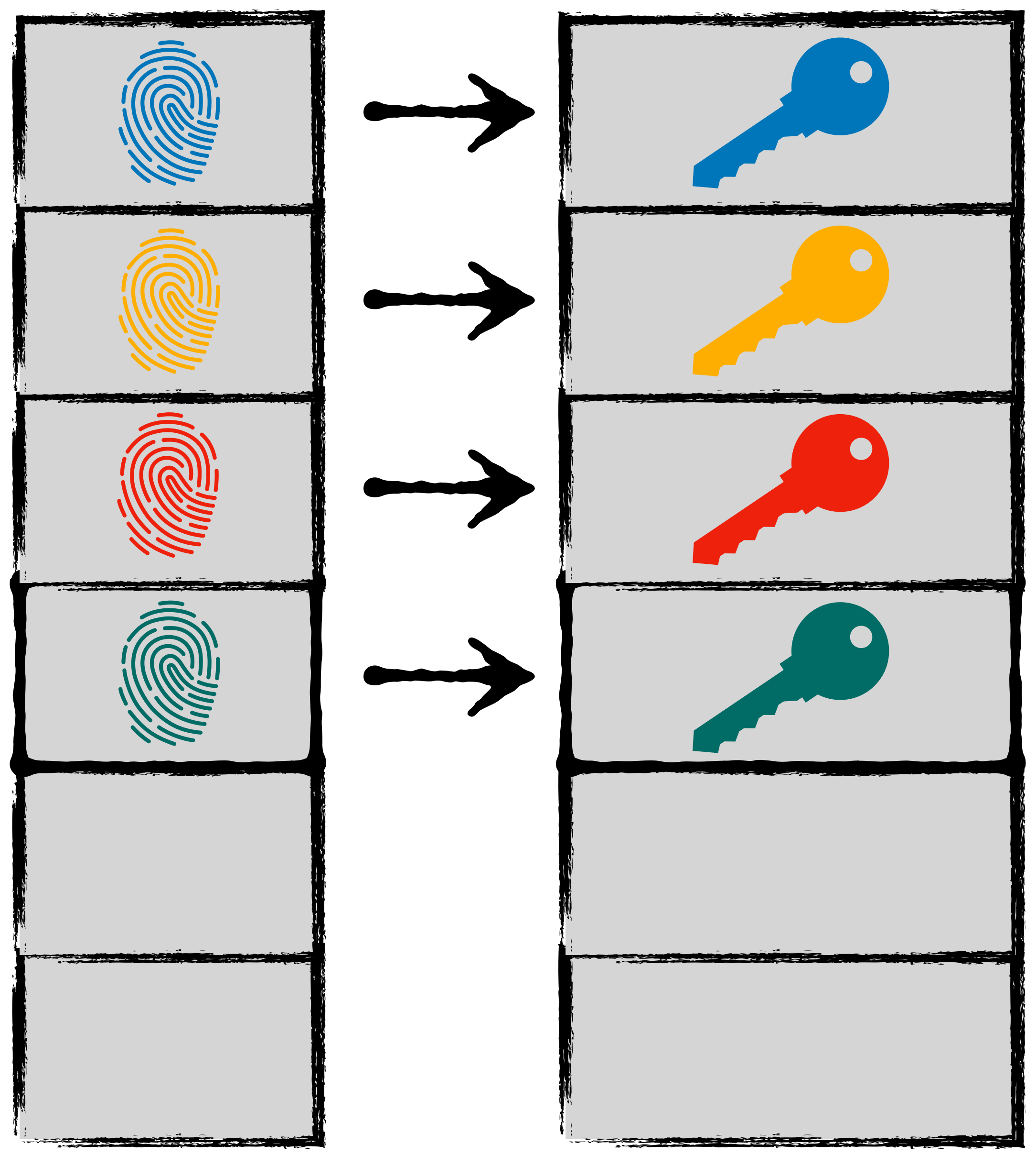


Fingerprint collisions can cause false positives

Adaptive filters employ variable-length fingerprints



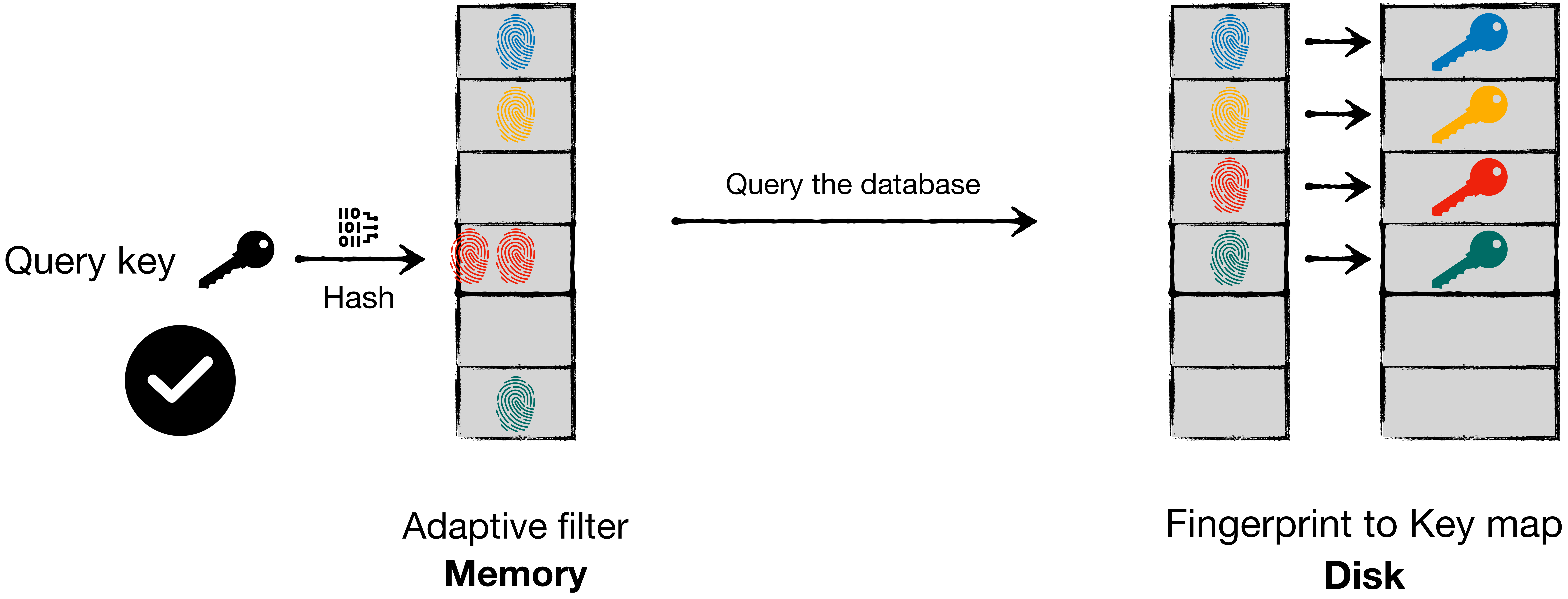
Adaptive filter
Memory



Fingerprint to Key map
Disk

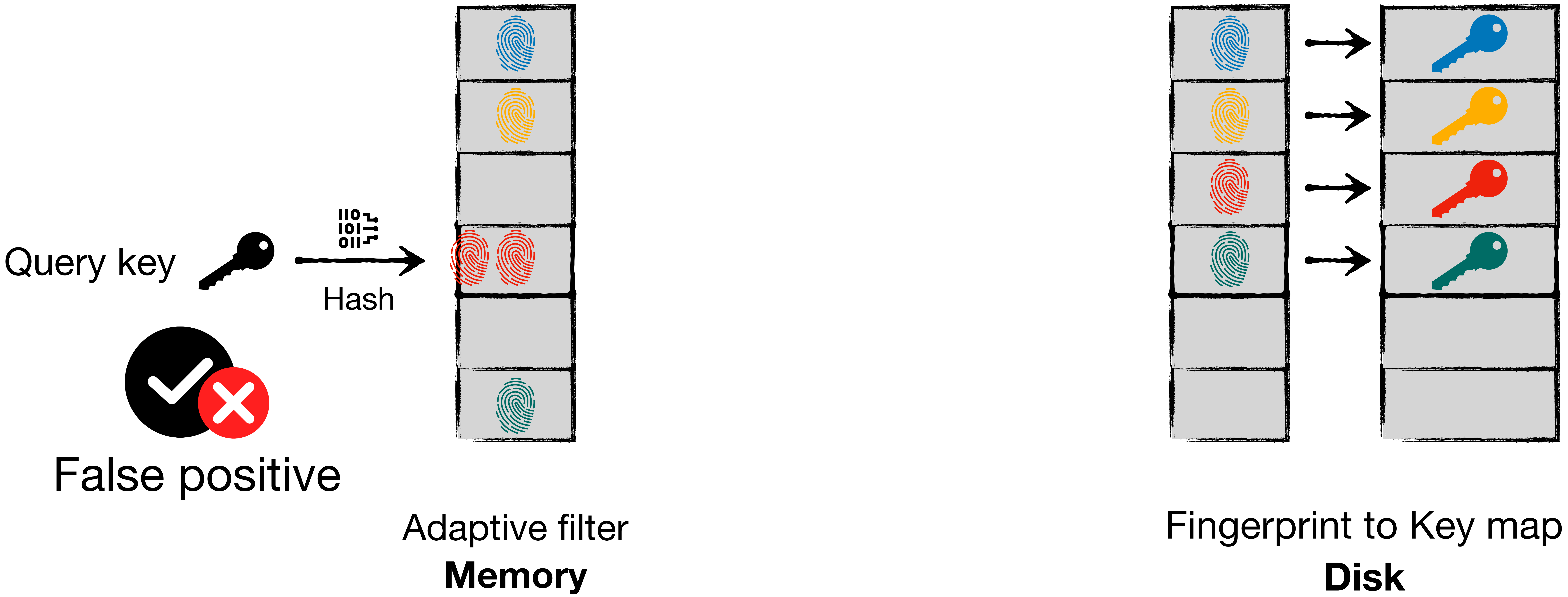
Fingerprint collisions can cause false positives

Adaptive filters employ variable-length fingerprints



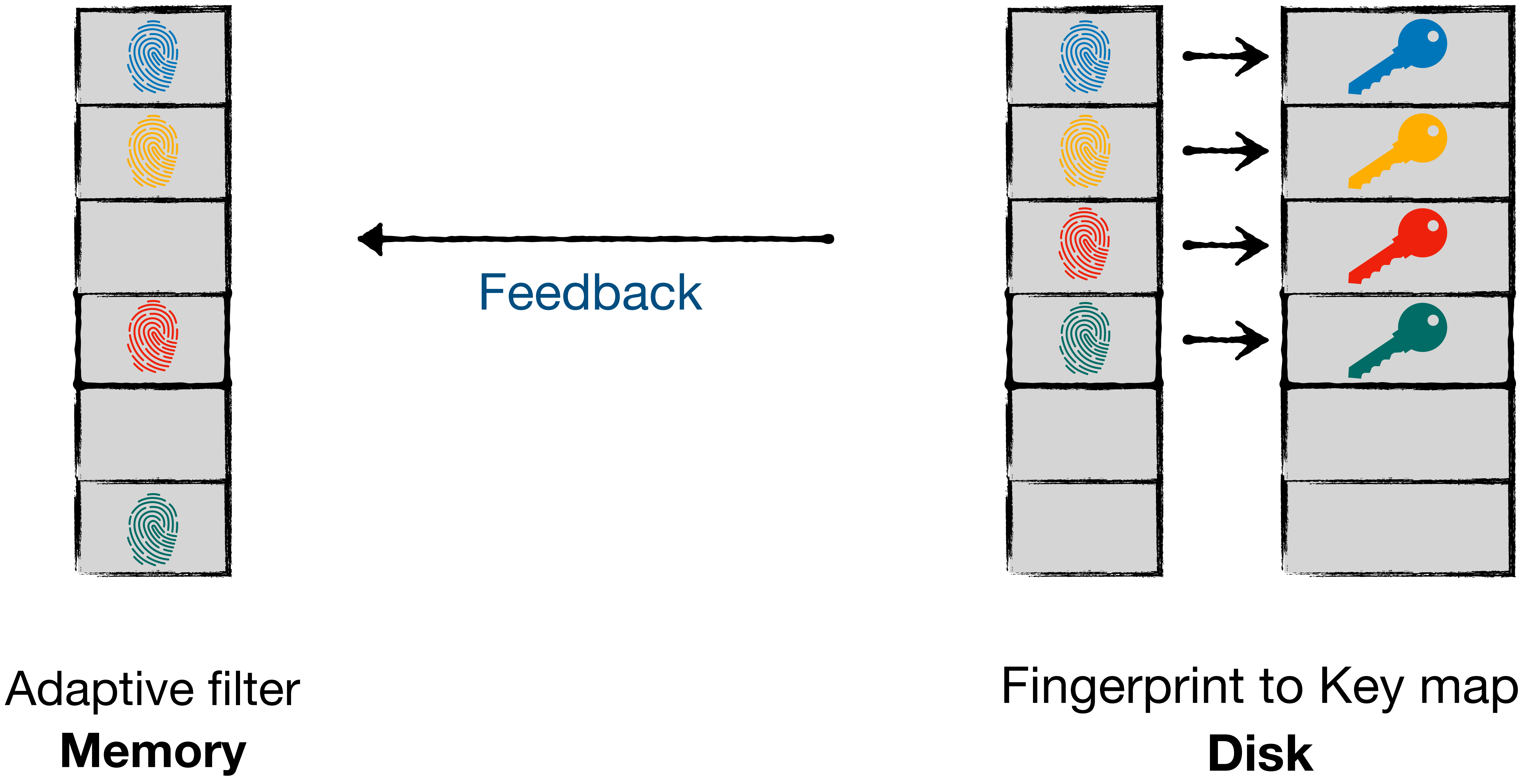
Fingerprint collisions can cause false positives

Adaptive filters employ variable-length fingerprints



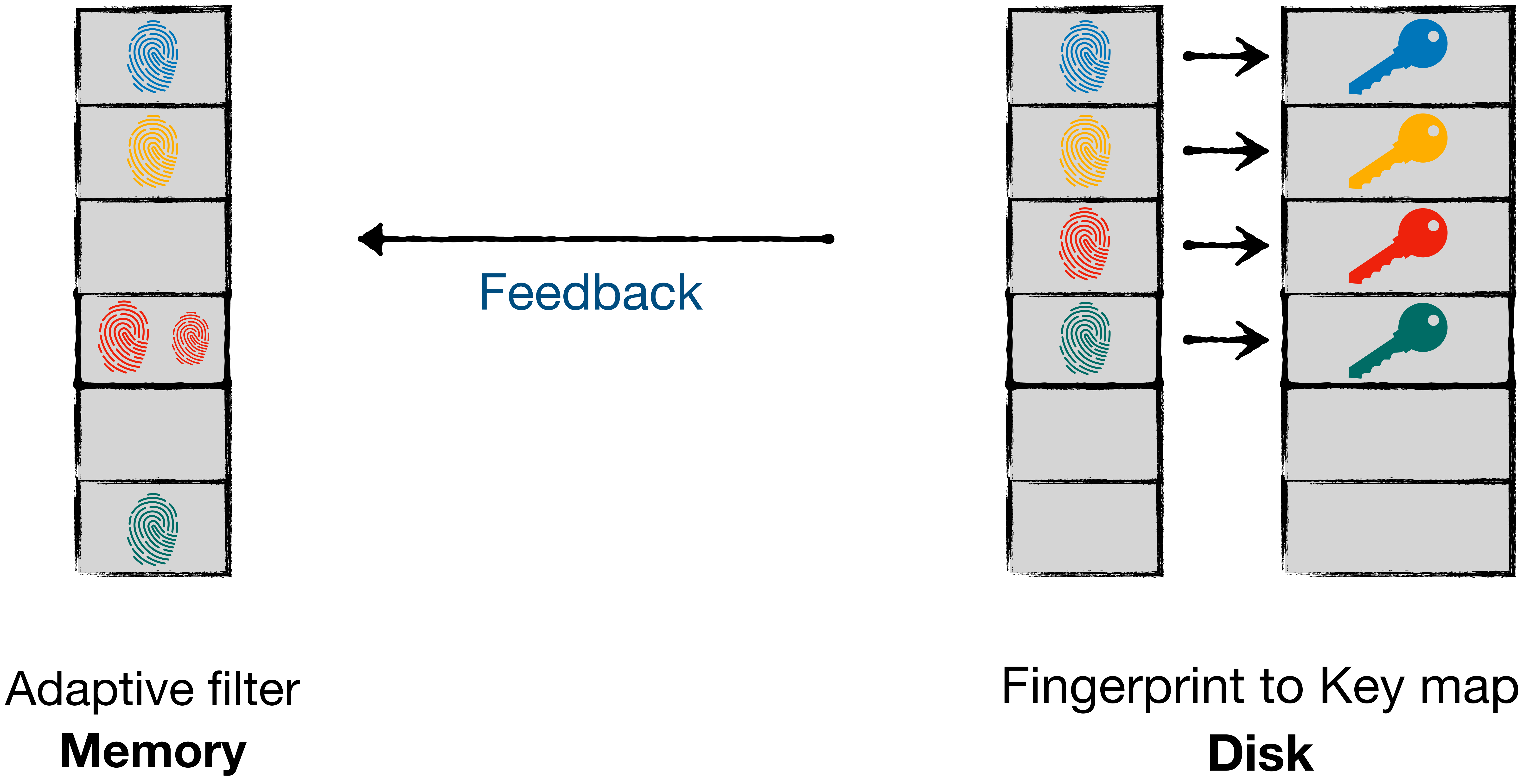
Fingerprint collisions can cause false positives

Adaptive filters employ variable-length fingerprints



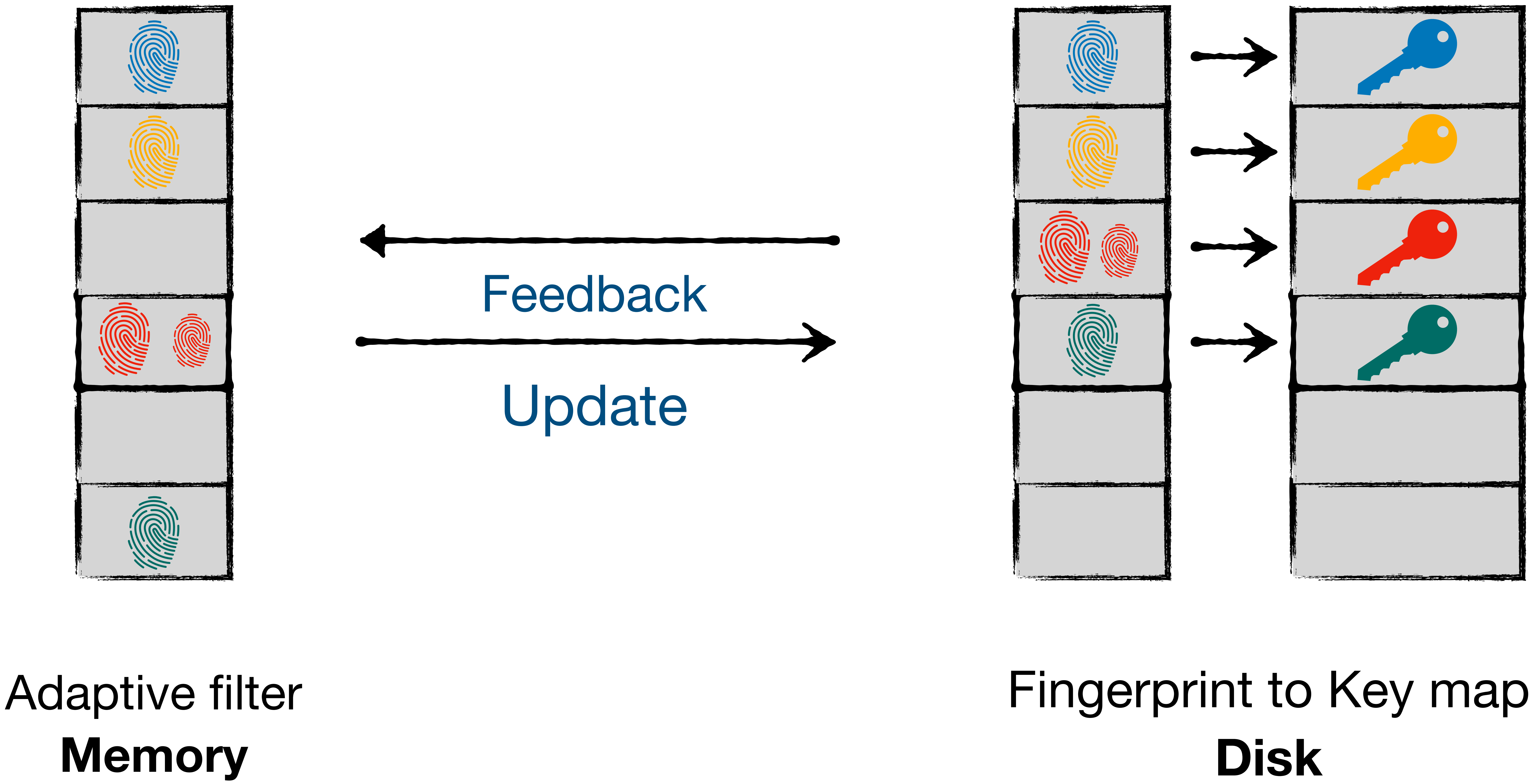
Feedback from the map can help fix the false positive

Adaptive filters employ variable-length fingerprints



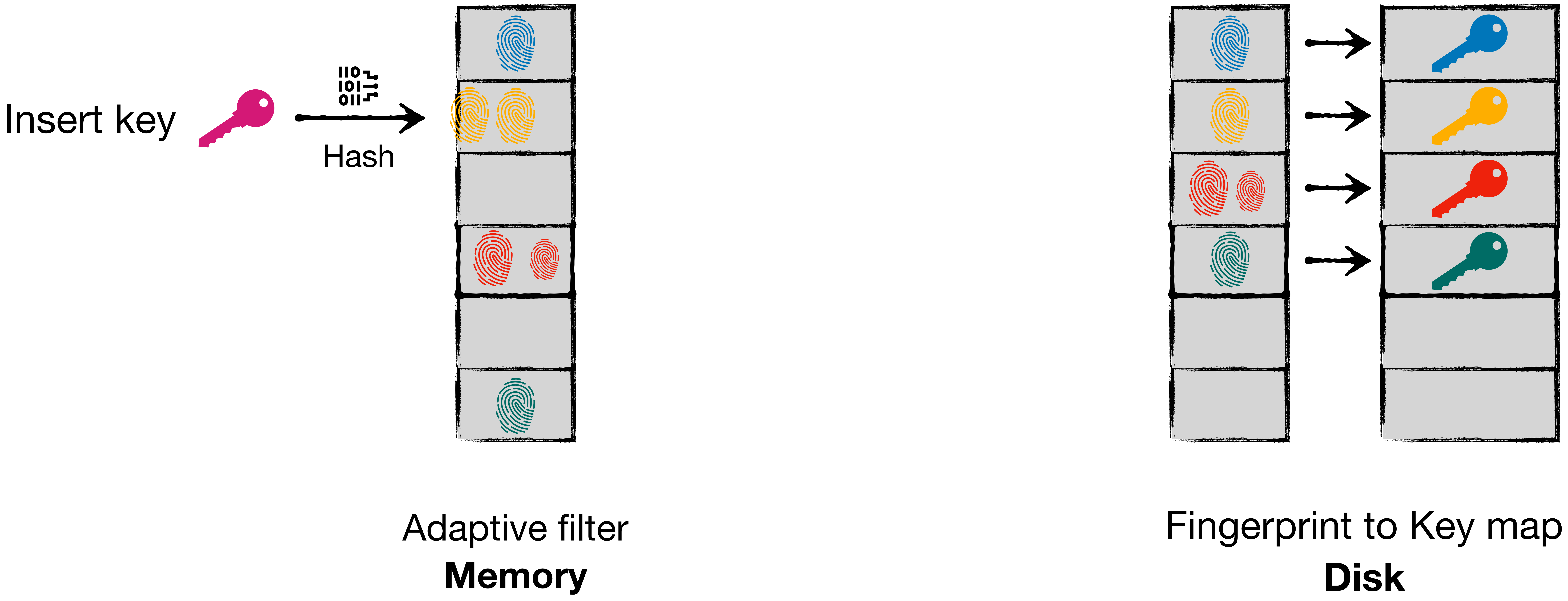
Extending the fingerprint of the existing key can avoid future false positives

Adaptive filters employ variable-length fingerprints

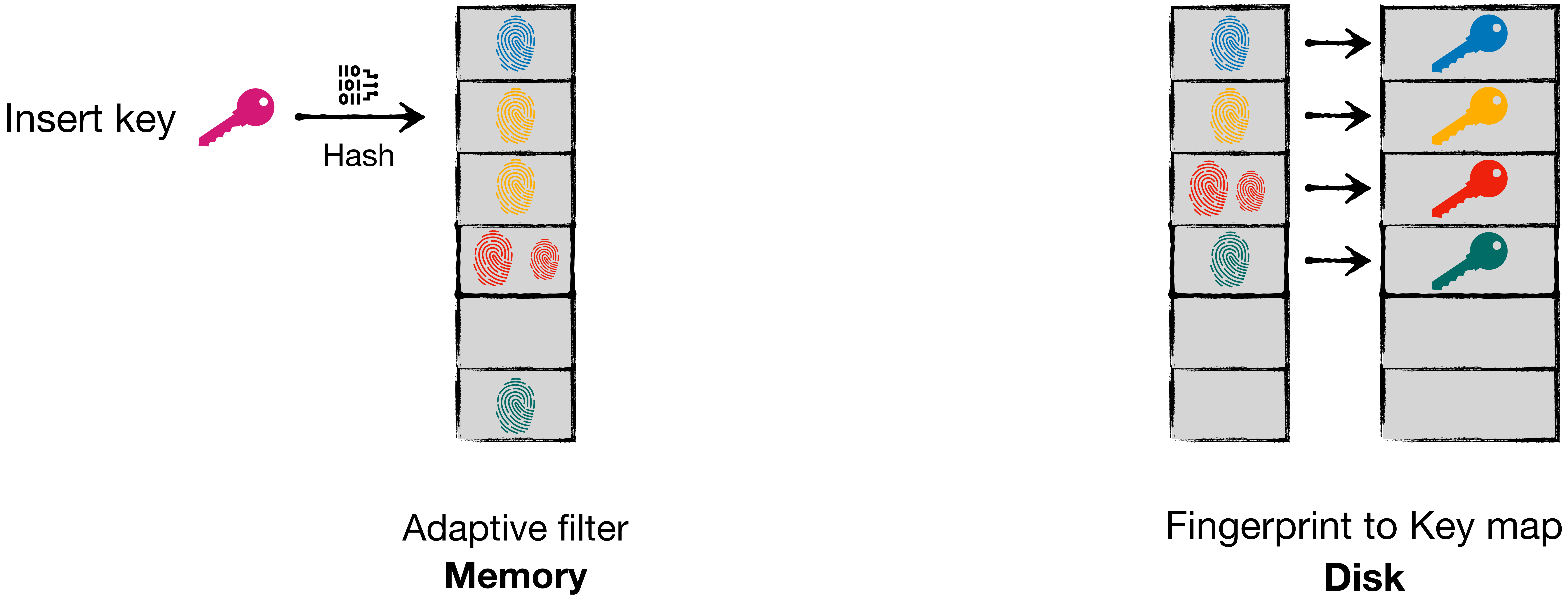


Fingerprint map is updated accordingly

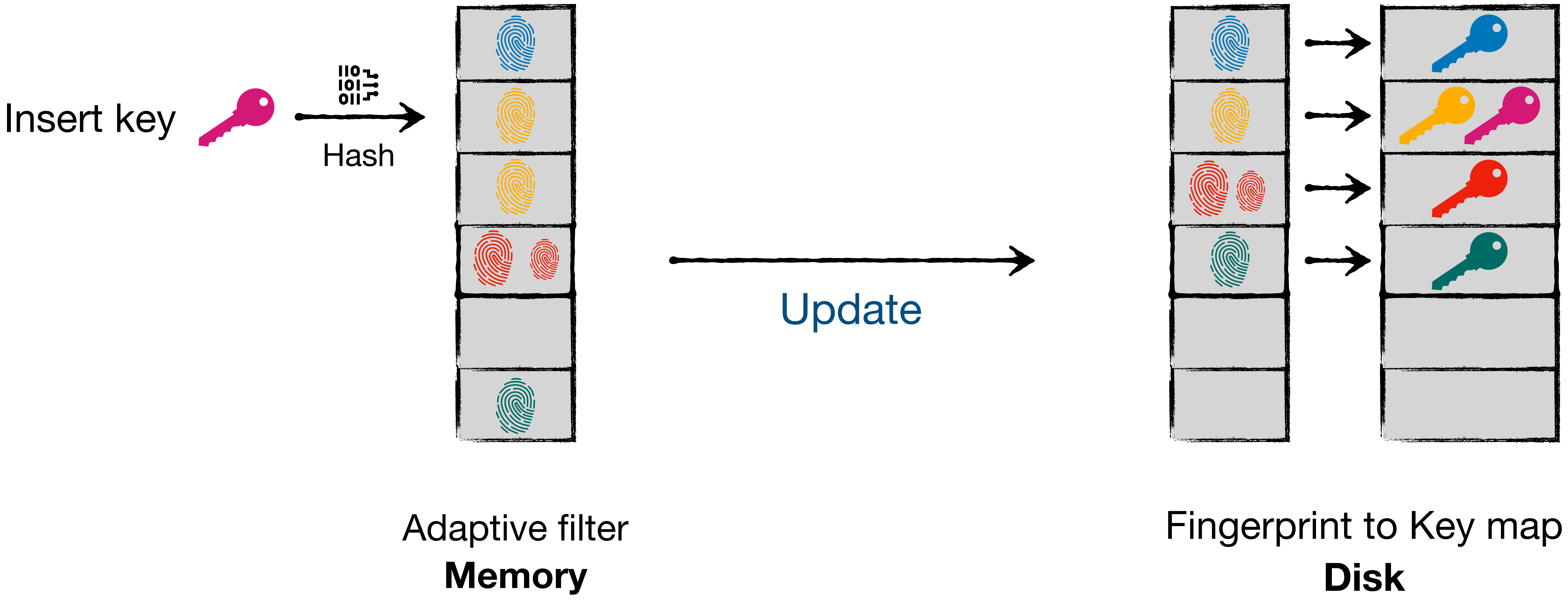
Adaptive filters employ variable-length fingerprints



Adaptive filters employ variable-length fingerprints

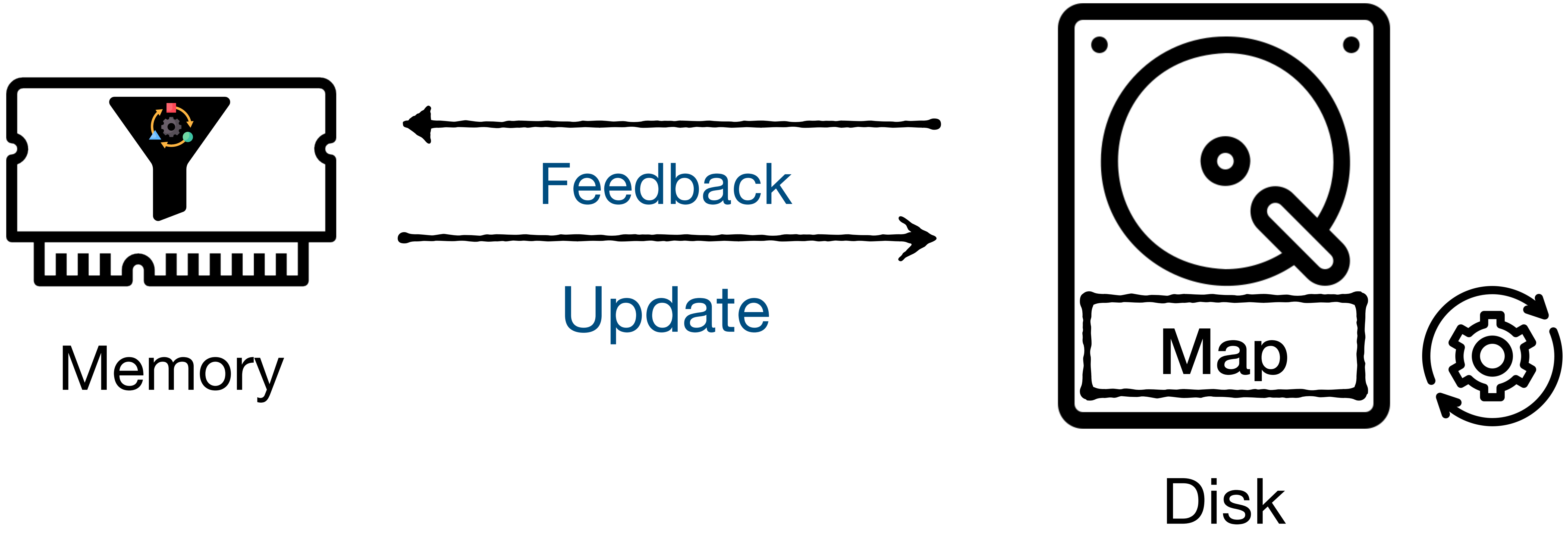


Adaptive filters employ variable-length fingerprints



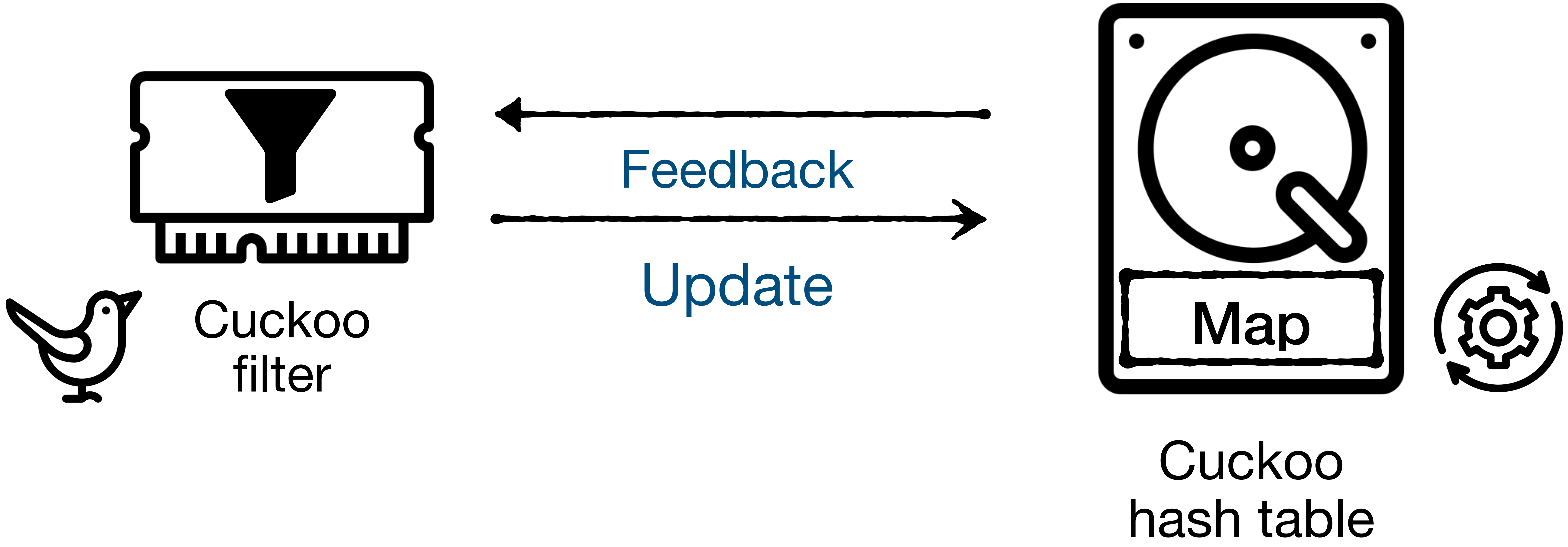
Fingerprint map is updated accordingly

Fingerprint map updates dominate the performance



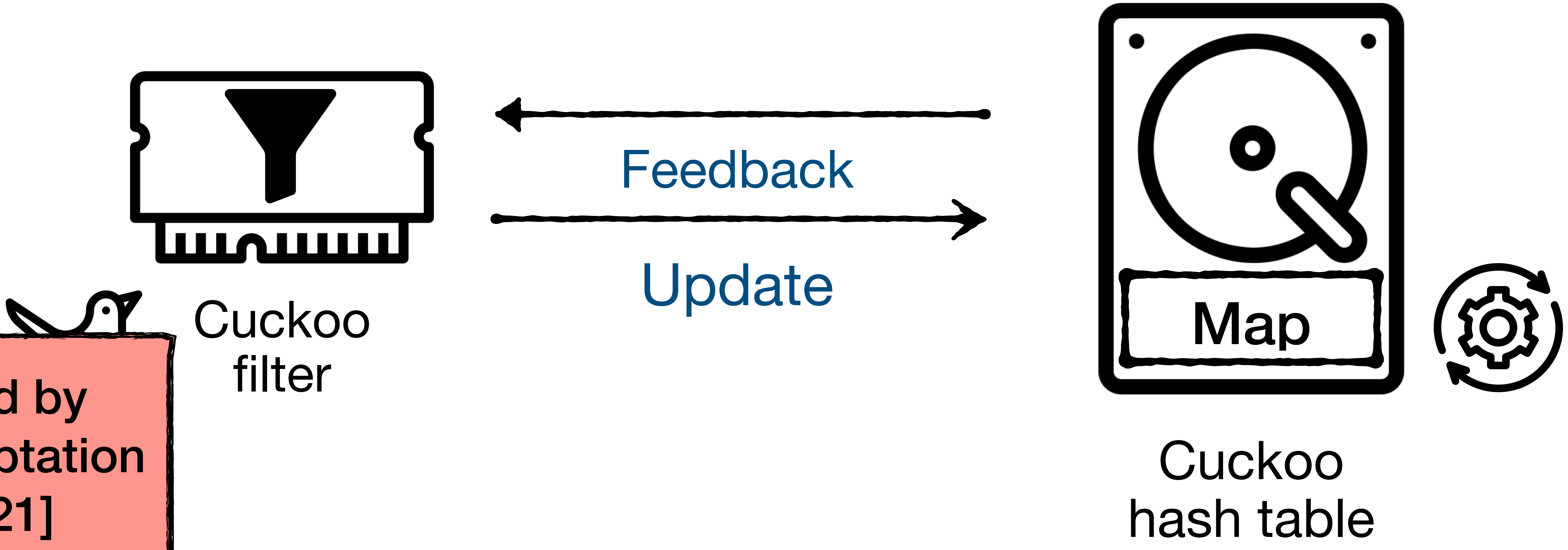
Minimizing the work in the map is crucial for the performance

Adaptive cuckoo filters [MPR+ 2020]



Adaptivity by moving fingerprints around

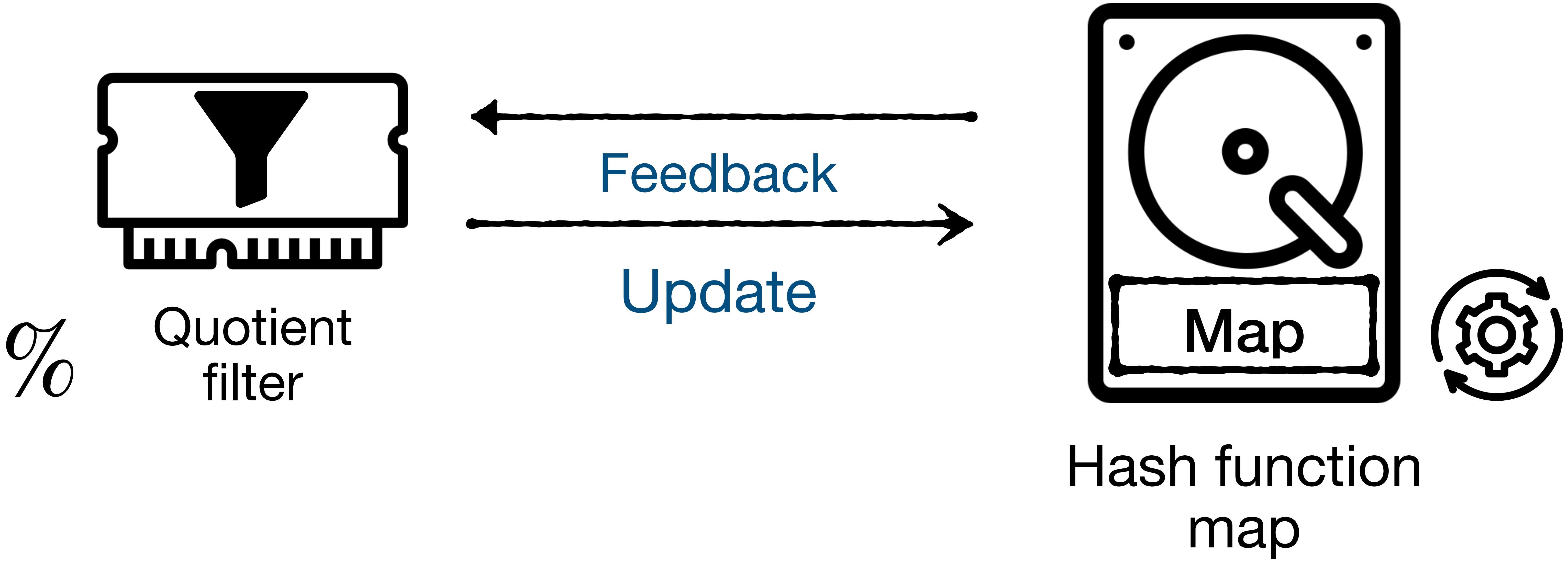
Adaptive cuckoo filters offer weak adaptivity



Can be attacked by identifying an adaptation loop [KMP 2021]

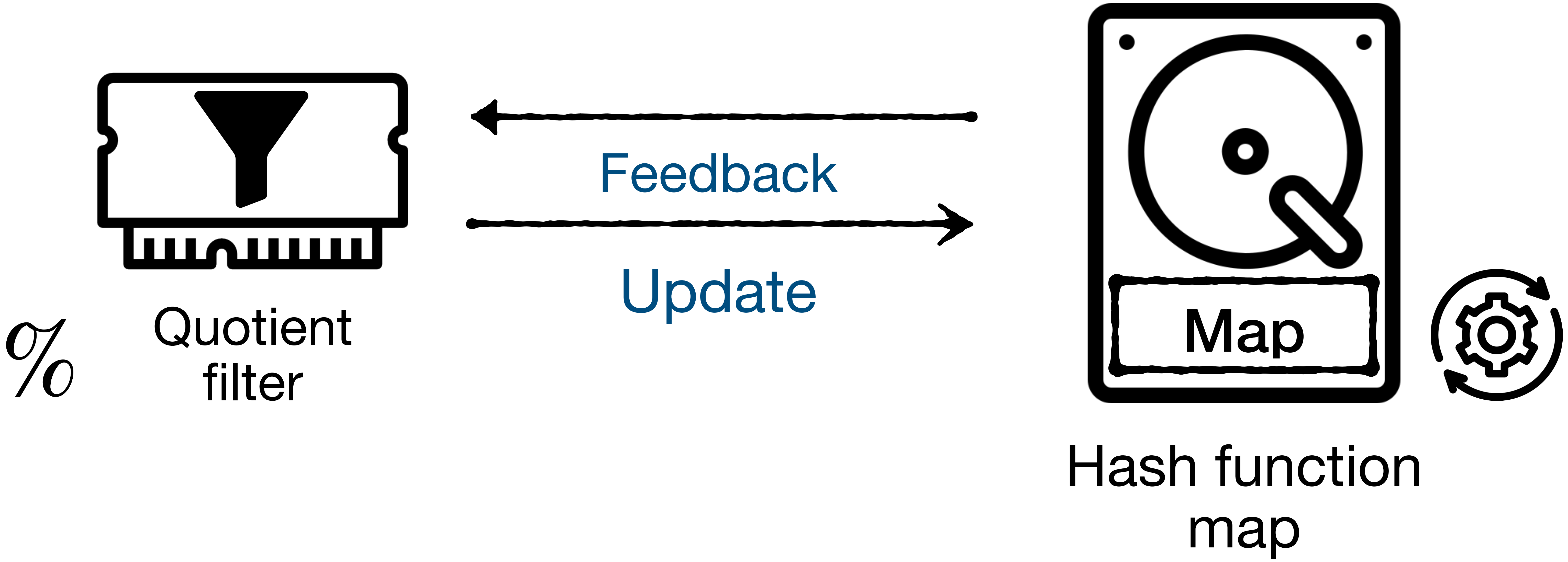
- Adaptivity by moving fingerprints around during insertions/queries
- Can forget previous false positives while adapting for new ones

Telescoping filters [LMS+ 2021]



Adaptivity by changing hash function during insertions/queries

Telescoping filters offer strong adaptivity



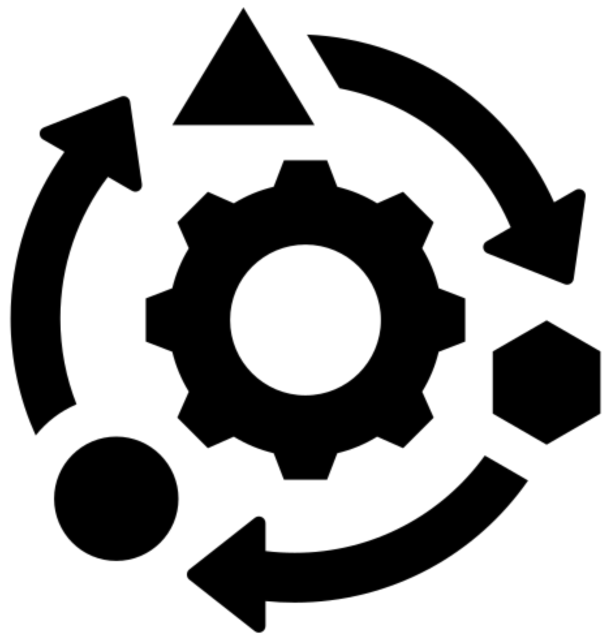
Adaptivity by changing hash function during insertions/queries

Hash map grows during adaptations (variable-length fingerprints)

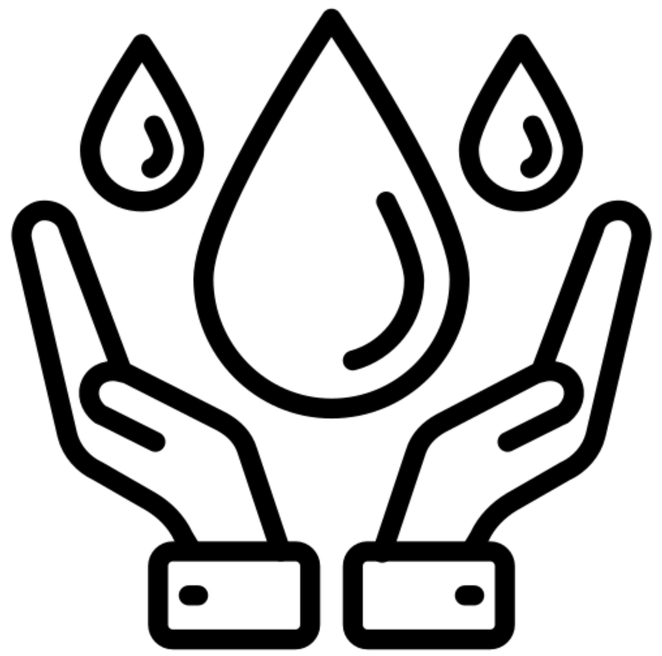
Does not forget previously learned fingerprints

Adaptive quotient filter [WMT+ SIGMOD 2025]

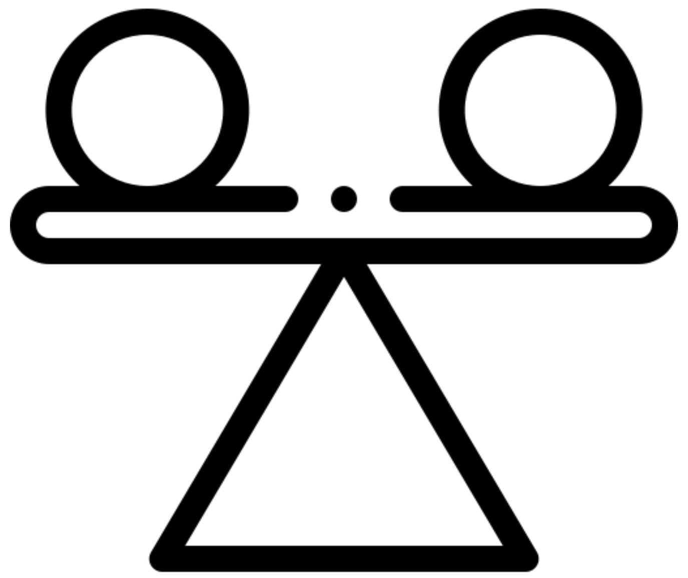
- Adaptivity by using variable-length fingerprints to avoid collisions
- Based on the counting quotient filter (CQF) [PBJ+ 2017]
- Matches the space lower-bound to lower-order terms
- 10X—30X faster than other adaptive filters (ACF, TF) for disk-based database benchmarks
- Up to 6X faster performance than traditional filters (QF, CF) for disk-based database benchmarks



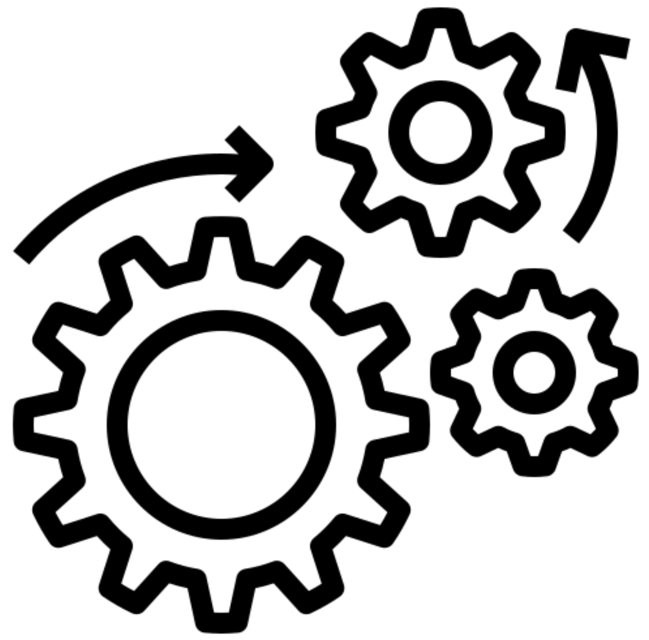
Adaptive quotient filter design



Preserves CQF
performance and features

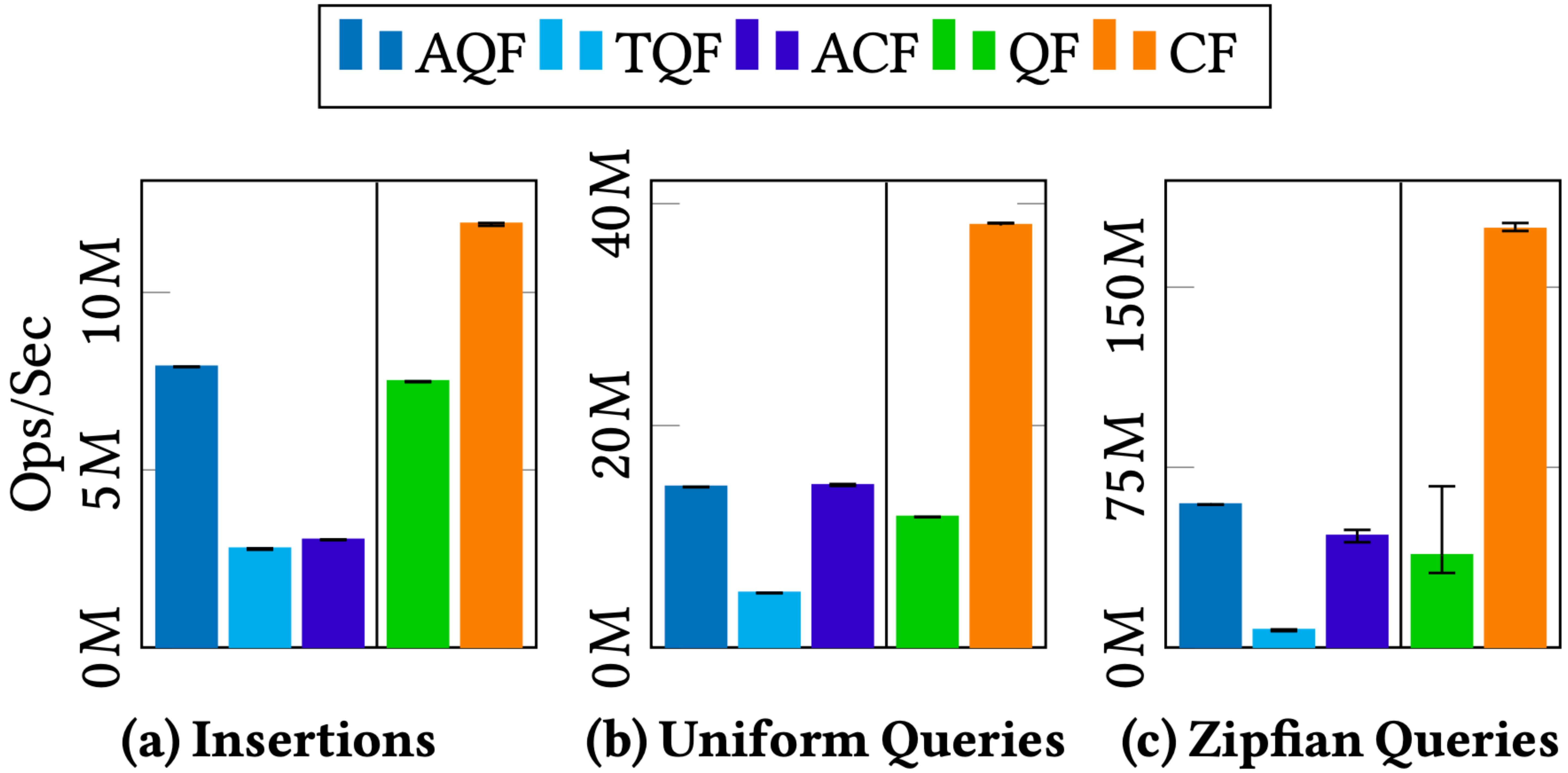


Stable reverse map
during insertions



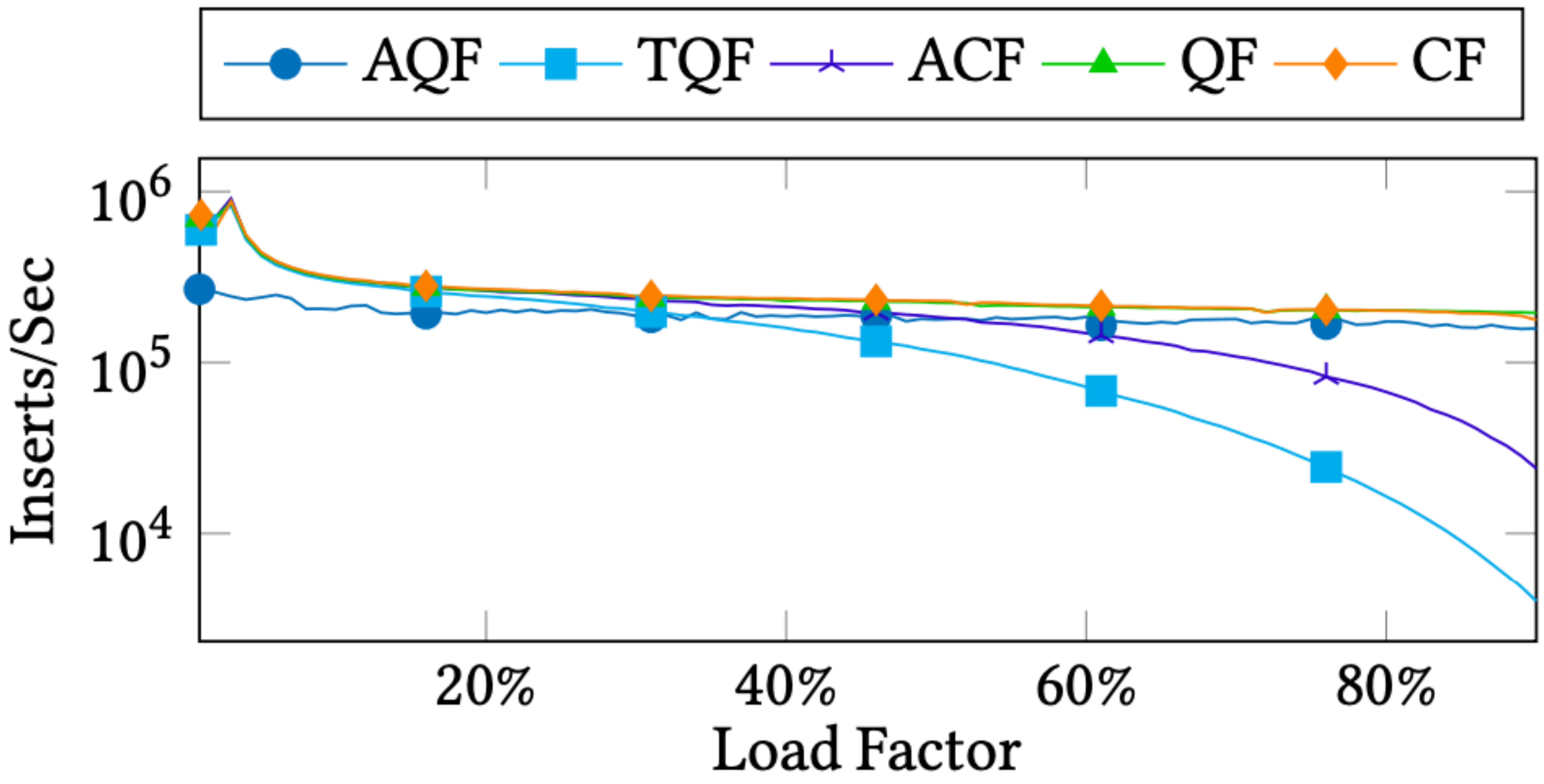
Supports dynamic
operations

Micro-benchmark performance



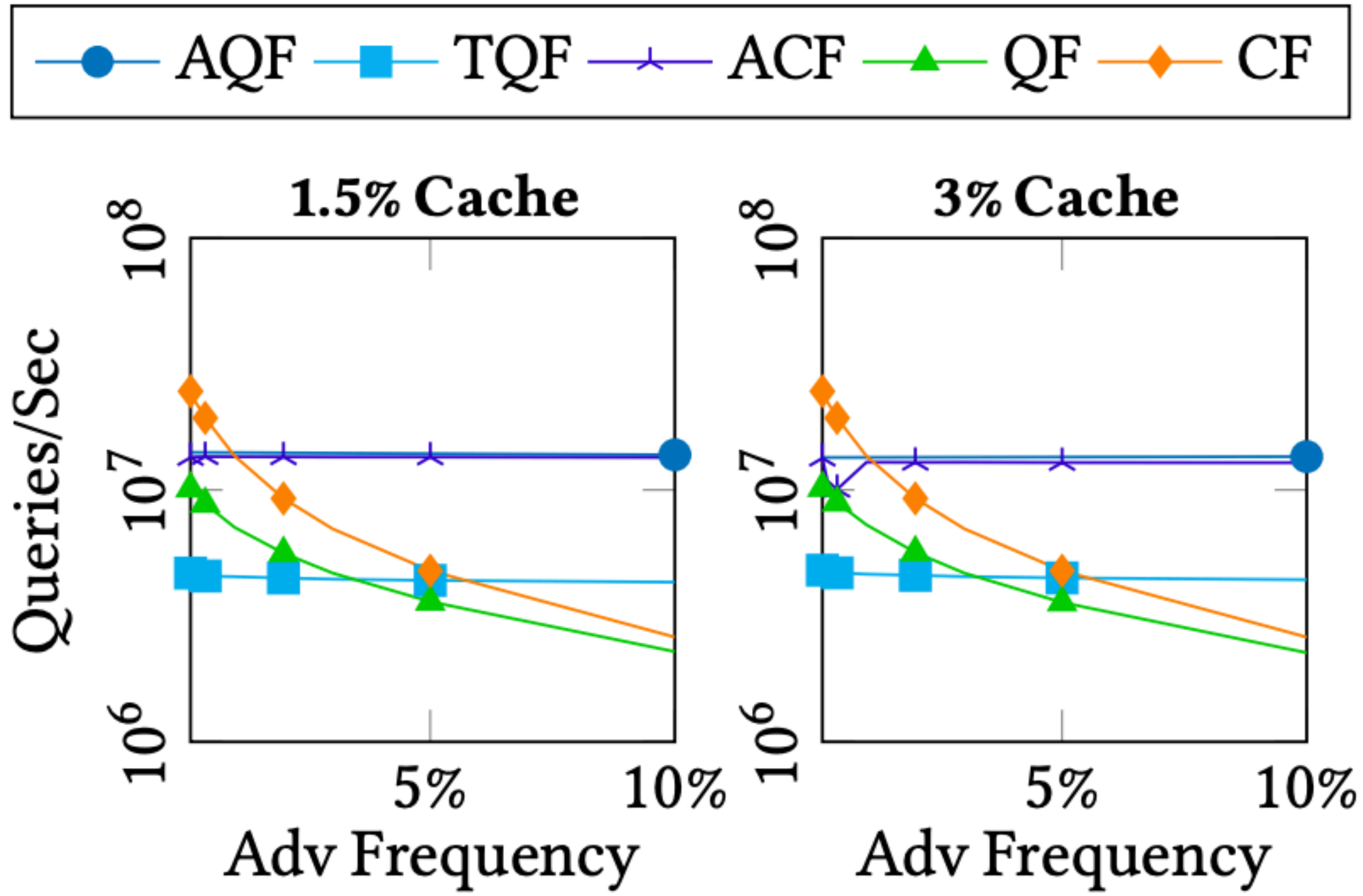
AQF has no overhead compared to the traditional CQF

Database insertion performance



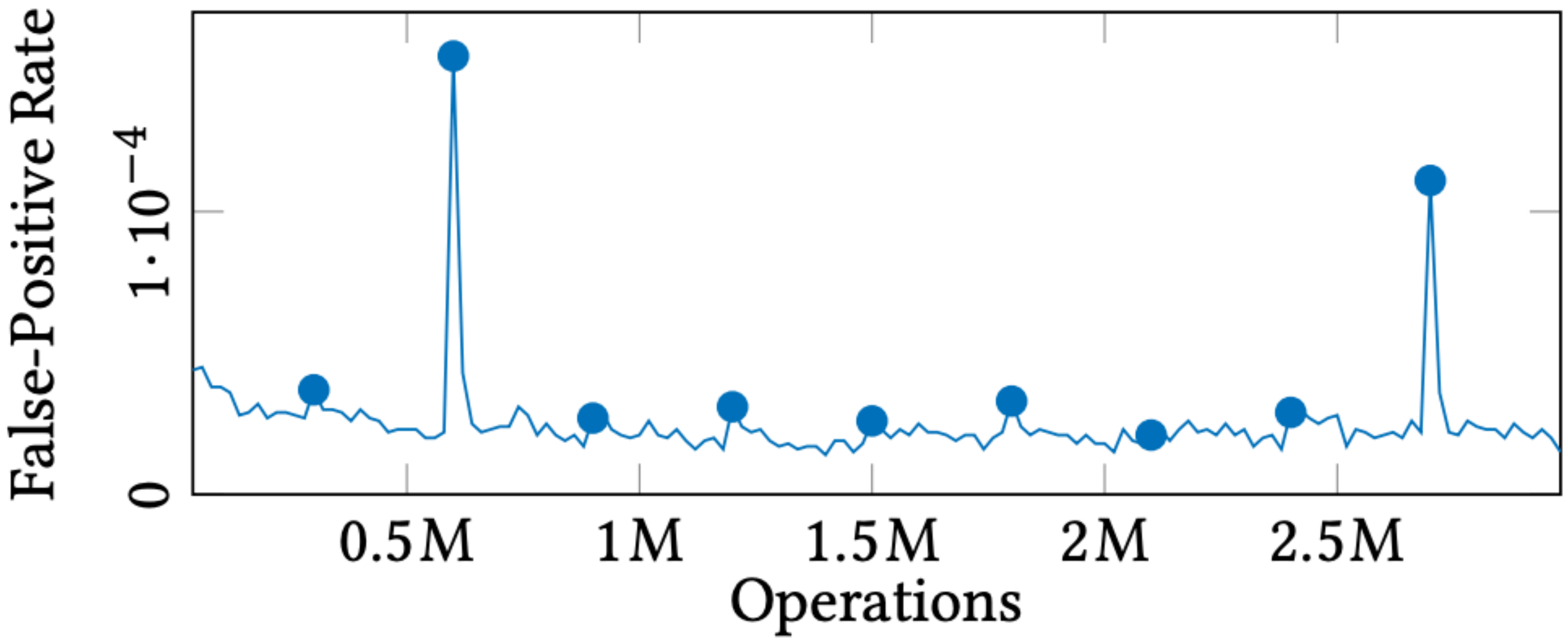
AQF performs similarly to QF/CF for database insertions
10X—30X faster than other adaptive filters

Database query performance



AQF up to 6X faster compared to QF/CF for database queries

Adaptivity rate on a churn workload



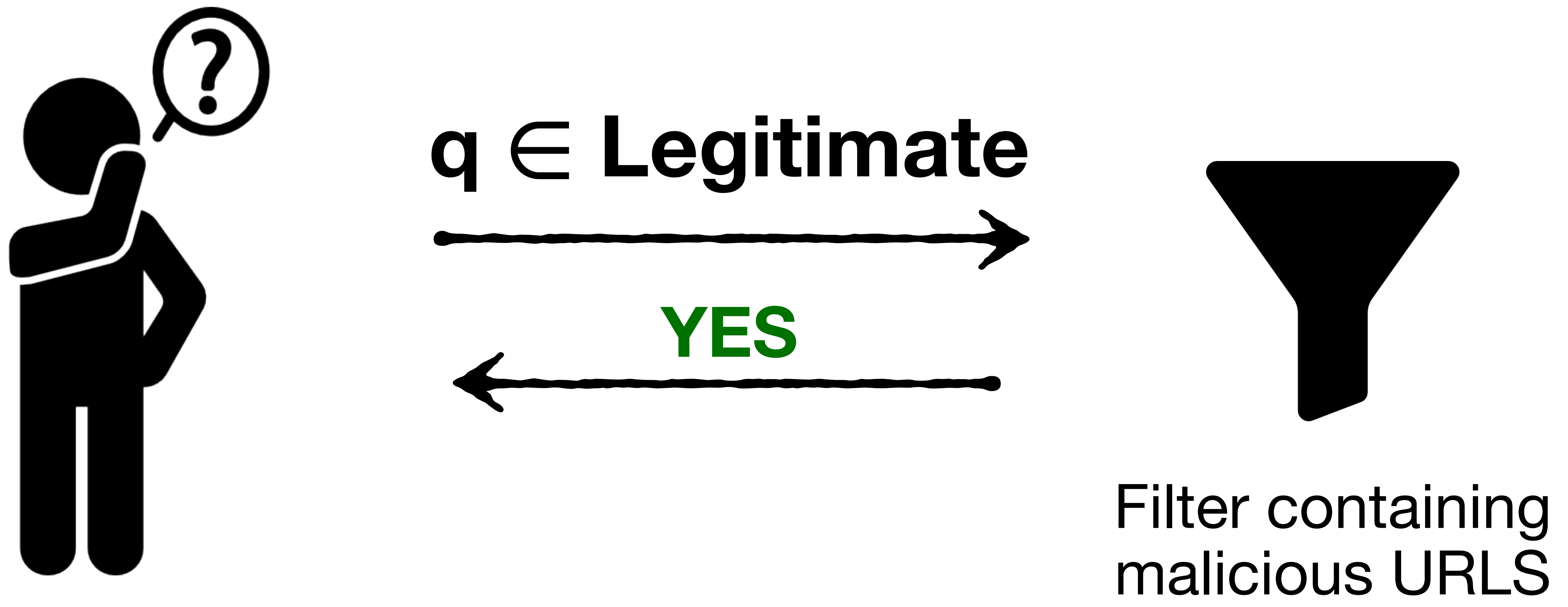
AQF adapts to new false positives almost immediately for churn workloads

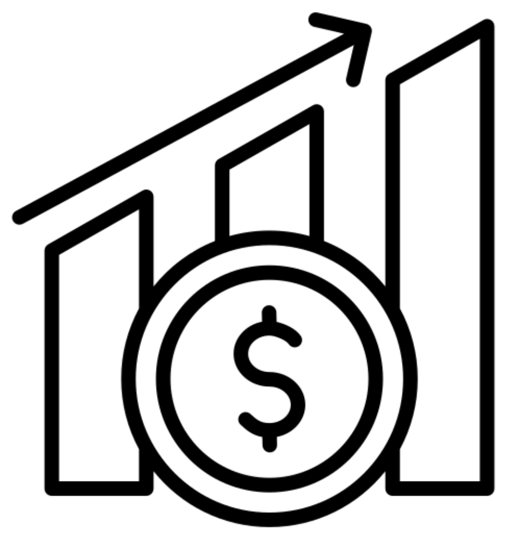
AQF offers even stronger guarantees compared to the broom filter [BFG+ 2018]

False positives can be really expensive

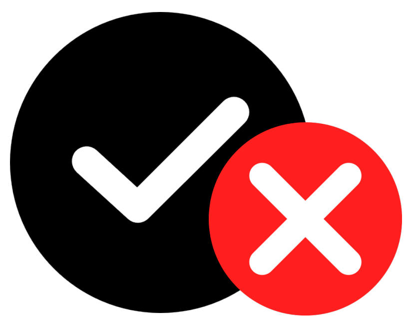
Malicious URLs

Legitimate URLs





A false positive can block critical URLs such as a voter registration webpage or emergency weather info



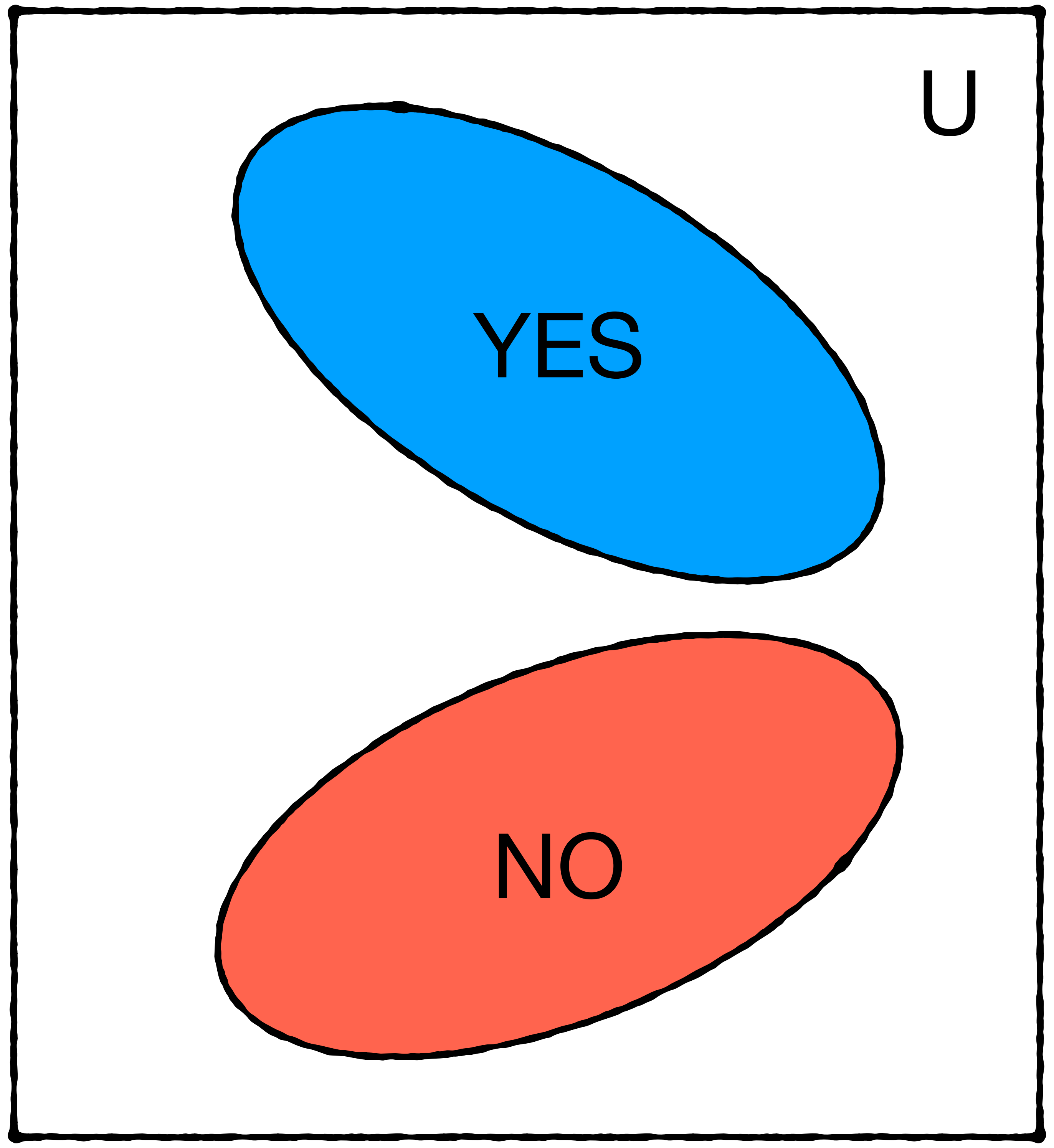
False positive

YES/NO list problem

if q ∈ YES, return **True** with probability 1

if q ∈ NO, return **False** with probability 1

Otherwise **False** with probability $> 1 - \epsilon$



Applications:

- Detecting malicious URL
- Certificate revocation lists
- De Bruijn graph traversal

Monotonicity is critical to support YES/O List problem!

Prior work considered each problem separately

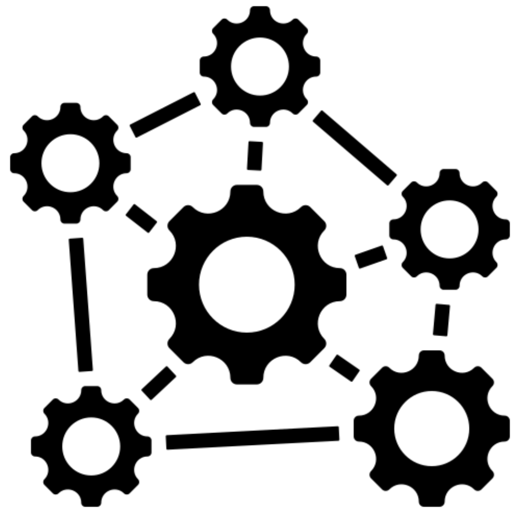
Purpose-built solutions

Bloomier filter [CKR+ 2004]

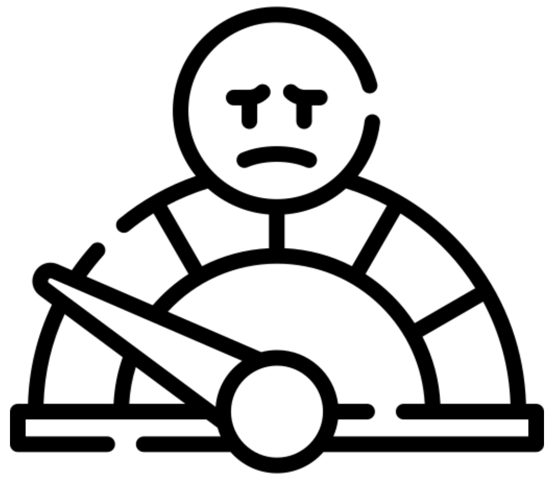
Cascading Bloom filter [TC 2009]

Static XOR filter [RSW+ 2021]

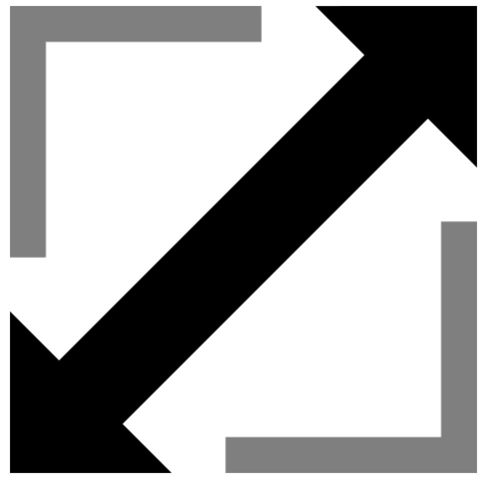
Seesaw counting filter [LCD+ 2022]



Complex design



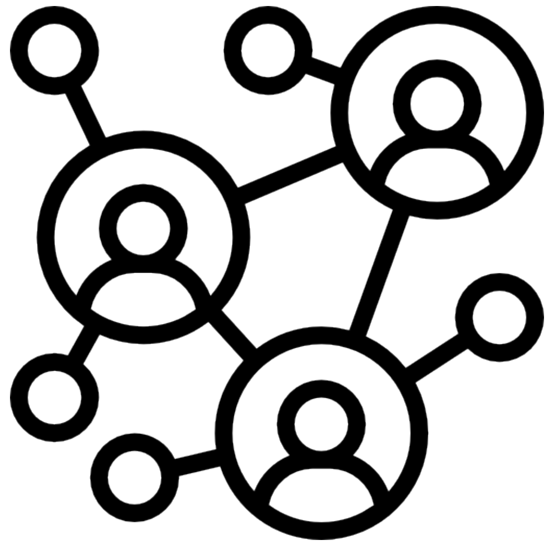
Low performance



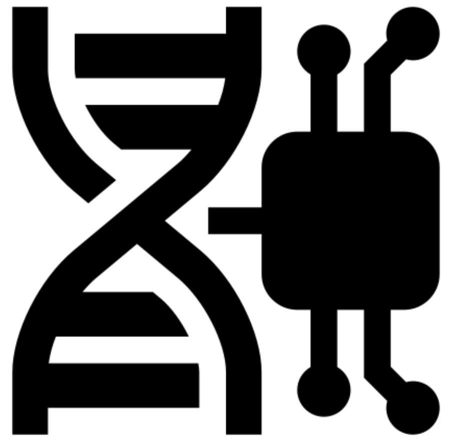
High space

Monotonically adaptive filters solve many problems

- Security; avoiding DOS attacks
  - Static YES/NO list
  - Dynamic YES/NO list

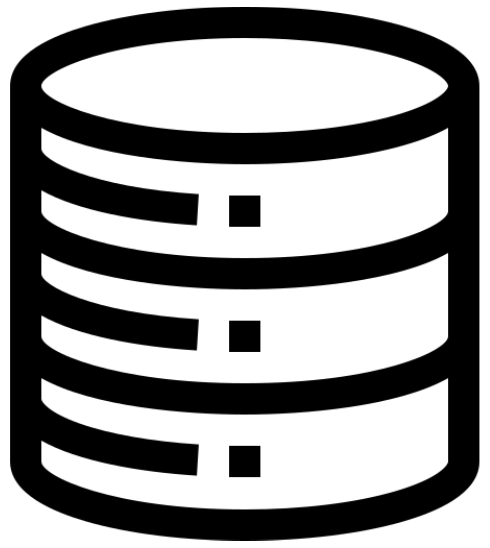


Networking



Computational biology

- Robust performance guarantees
  - Skewed query distributions
  - Adversarial queries



Databases

Takeaways

- Adaptability is a critical to achieve robust performance in the context of skewed/adversarial workloads
- Monotonically adaptive filters can help address challenges across applications
- We need to redesign traditional applications in the context of newer guarantees and API offered by adaptive filters

Acknowledgment: All icons in the talk are taken from https://www.flaticon.com/