Bloom Filters

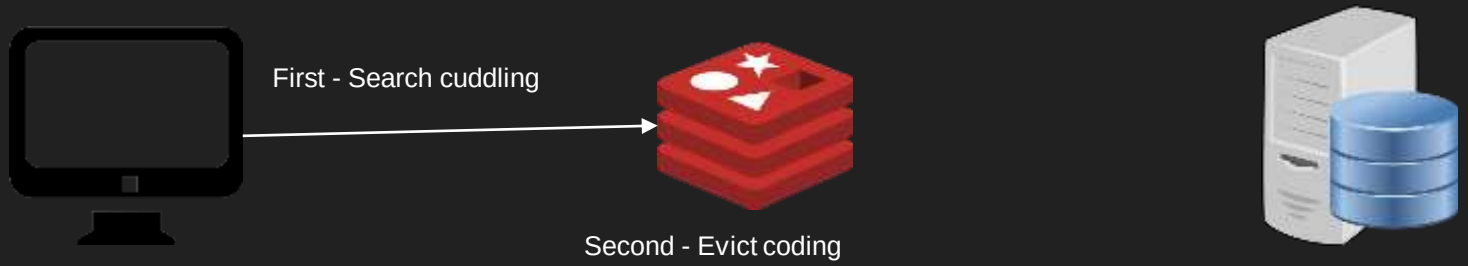
Background

Bloom Filters are a data structure used to approximate the contents of a set of elements. They are known as a probabilistic algorithm.

While Bloom Filters may be incorrect and say that an element is in the set (false positives), they are always correct when saying an element is not in the set (no false negatives).

Bloom Filter use case - cache filtering

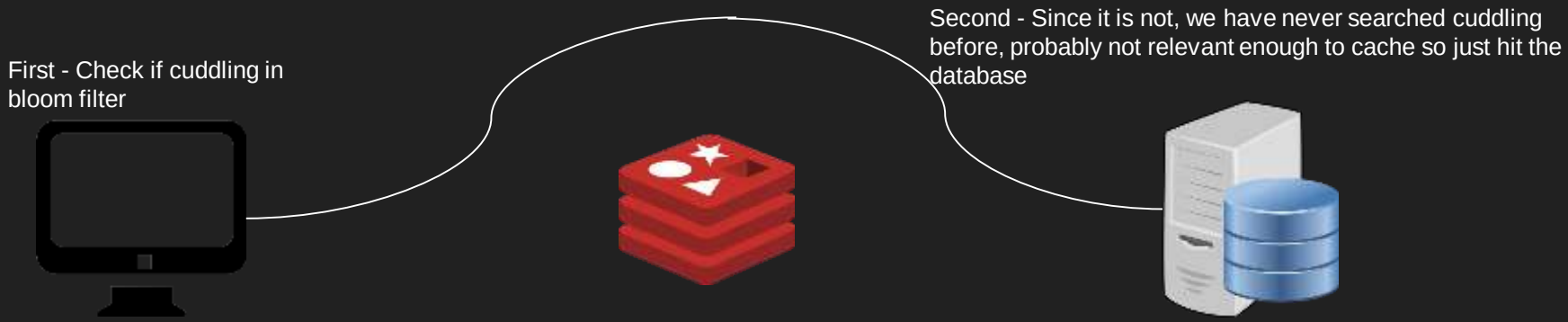
Recall: Most cache has to evict one piece of data in order to load in another one.



Bad! We never meant to search cuddling, it was just an autocorrect! Wouldn't it be better to know that we've never searched cuddling before to help show the cache that it shouldn't evict a more relevant entry?

Bloom Filter use case - cache filtering

Recall: Most cache has to evict one piece of data in order to load in another one.



Bloom filters allow us to see if we have seen a given term before to estimate whether it is relevant enough to be cached!

Bloom Filter use case - SSTables

Recall: When reading from an LSM tree based storage engine, we must check all the SSTables in order from newest to oldest until we find the key that we want.

Searching for key jordan: we can quickly skip over binary searching SSTables 1 and 2 because their bloom filters (probably) return “no” for jordan

SSTABLE 1

aidan: 13
bob: 2
zachary: 25

SSTABLE 2

adam: 4
charlie: 17
philip: 21

SSTABLE 3

daniel: 5
jordan: 21
zachary: 20

Bloom Filter Algorithm

Start with array of m bits initialized to 0s

Bloom Filter Algorithm

Start with array of m bits initialized to 0s

Choose k different hash functions:
if key “poop” is in the set get the
result of each of the k hash functions
for poop mod m

h_1(poop) % m = 0
h_2(poop) % m = 2
h_3(poop) % m = m-1

Bloom Filter Algorithm

Start with array of m bits initialized to 0s

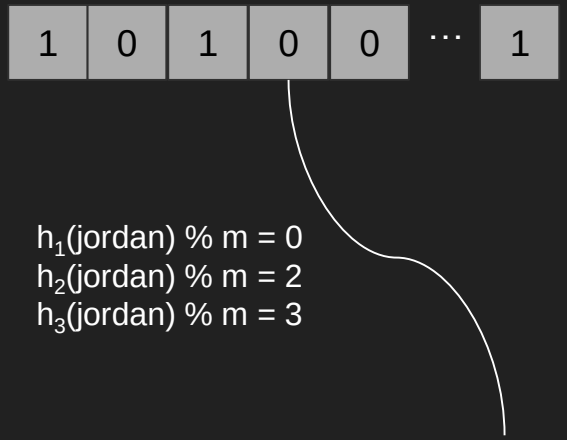
Choose k different hash functions:
if key “poop” is in the set get the result of each of the k hash functions for poop mod m

h_1(poop) % m = 0
h_2(poop) % m = 2
h_3(poop) % m = m-1

Change all the bits corresponding to the results of the previous equation to 1 in the Bloom Filter

Bloom Filter Algorithm - Continued

Bloom filter for s = {poop}



Get hash results for key = jordan

h_1(jordan) % m = 0
h_2(jordan) % m = 2
h_3(jordan) % m = 3

Since filter[3] = 0, jordan cannot be in the set!

Bloom Filter - Caveats

- As we add more elements to the set, more of the bits are going to start switching to 1s
- This means that sometimes we will get unlucky and it will seem like certain keys are in a set even though they are not (false positives)
- Occasionally it makes sense to reset a bloom filter to all 0s in order to lower the rate of false positives, at the cost of losing the data already in the filter

Conclusion

Bloom filters are a very interesting and clever algorithm based on hashing that can be useful both in competitive programming and systems design. They are employed in many LSM based storage engines and are largely responsible for their respectable read performance compared to B-Trees.