CS 224: Advanced Algorithms Spring 2023

Lecture 12 — March 13, 2023

Prof. Prashant Pandey *Scribe: Saurabh Raje*

1 Overview

In this lecture we discussed filters, primarily bloom filter and single hash filter. Specifically, the space complexity as a function of acceptable false positive rate was compared. Finally, some known challenges in the two filters were discussed. These shall motivate future lectures that discuss how to design more malleable filters.

2 Comparing filter with hash table

A filter with false positive rate of ϵ requires at least $n \log \frac{1}{\epsilon}$ bits to store. In comparison, a hash-table requires $\Omega(n \log |U|)$ bits to store, and gives no false positives.

3 Bloom filter design

A Bloom filter [1] simply an array of bits, that represent what elements of the universe exist in a collection. The position of the bits is used to encode this information for specific elements. A hash function maps a given element of the universe to a fixed number of positions in the bit-array. An "insertion" of the element sets all of these positions to 1. A lookup for the same element would hence reduce to the AND operation over the bits at those particular positions. By above construction, the filter guarantees that there will be no false negatives. This is because once bits are set to 1, no operation "unsets" the bits. Therefore, if data was added to the filter, the filter possibly can't output that the data is absent. Bloom filter is parameterized by two variables:

1. m: size of the bit-array.
2. k: number of positions that the hash function gives for every element in the universe.

The space required by a bloom filter is $1.44n \log \frac{1}{\epsilon}$. Given N items and ϵ error rate,

$$k = \frac{m}{n} \ln 2 \tag{1}$$

Deletion from filters is very difficult. It can be somehow managed with 4-bit counters. If we naively set the bits to zero, essentially doing the opposite of insertion, the bloom filter will have false negatives. This is undesirable, hence deletion is typically not supported in bloom filters.

3.1 Modifications to bloom filters

Bloom filters can't be resized, merged with another filter. Furthermore, you also can't delete items from a bloom filter with a single bit per position. This is fundamentally because the data stored in the bloom filter can't be retrieved in its original form. There is no way to get all the elements of the universe that exist in the bloom filter. Furthermore, resizing the filter is impossible because any keys added before growing the filter won't map to the same positions after growing the filter.

4 Single hashing filter

This method does not use a bit-array, instead it is backed by a hash-table. Any key from the universe is mapped to a P-bit value using a lossy hash function. The collection of all P-bit values is now stored in a hash-table.

The false positives in this setup come from the fact that two different keys from the universe, might be hashed to the same P-bit vector.

$$\mathbb{P}\left(x \; and \; y \; collide\right) = 2^{-p}. \tag{2}$$

Note that this does NOT depend on $|U|$. After inserting n items, the probability of collision is $<= \frac{n}{2^p}$. By definition, this is also the false positive error rate bound. The space required for this structure is np bits. Space required by bloom filter would have been higher.

5 Typical cache line access latencies

L1 latency is 3ns, L2 latency is 5-6ns, L3 is 12-20ns and DRAM is 60ns.

6 Command to check cache sizes

On any linux system, the *lscpu* command will show the hardware topology, along with the capacities of the different volatile memory in the chip.

References

[1] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, jul 1970.