Liam Murray

Professor Miller

CSE13s

November 20, 2022

Assignment 6 Writeup: The Great Firewall of Santa Cruz

Question One:

In regards to the Bits Examined Per Miss (BEPM), it appears that as the size of the bloom filter decreases, the BEPM increases. I ran a series of test with following sample input:

> "Hello Good Citizens of the Santa Cruz. We need liberty. Government is bad. The accad is evil and unjust. Adin these peanuts. We must band all arm and feet together as one."

Each test, I decreased the size of the bloom filter by a factor of two. As I went, I noticed that the BEPM increased gradually from ranges 2^21- 2^17, and began to increase more rapidly as the size got smaller. For example, the difference between bloom filters of size 2^19 and 2^18 is 0.047605, whereas the difference between 2^15 and 2^14 is .625. This is an increase of over 1,000% within the span of 4 decreases in magnitude. Overall, as bloom filter size decreases, Bits Examined Per Miss increases.

Question Two:

The data in reference to hash table lookups and bloom filter size is very similar to the results discussed in question 1. Using the same method, the data reflects a gradual increase in hash table lookups corresponding to the decrease in bloom filter size. Using the same benchmarks, the difference between hash table lookups with a bloom filter of size 2^19 and 2^18 was 0 (both 17), whereas the difference between 2^15 and 2^14 was 7 (38 and 45 respectively).

This shows the same trend as the bits examined per miss, in that as the bloom filter size decreases, the bloom filter lookups increase.

Question 3:

After trying several different unique cases, the usage of the “Move to front” search option has proven itself to be a helpful optimization tool for some cases, however not necessarily generally applicable. The three types of input I used in each trial was designed to test how MTF would affect unique types and data trends.

The first input I tried was random, speech like text that resembled what an actual conversation might look like. In these cases, MTF helped, but often the difference was negligible, usually amounting to a decrease of one or two links. This is likely due to the fact that certain, more naturally common words got moved to be more accessible, however they were still surrounded by less common words which increased the overall number of links traversed.

The second data type I tried was a list of the same word. This was designed to find the best possible case for MTF. In these tests, one of two things happened. Either, the difference was massive (a test with the word “Jabar” resulted in a decrease of almost 50%), or there was no difference at all. This is determined by where the word is originally in the hash table. If the word starts at the top of its linked list, then MTF essentially does nothing. However, if it is lower down on its linked list, MTF can significantly decrease the lookup times.

Finally, the last case I tested was all unique words. In this case, MTF did more harm than good. Usually the difference between MTF true and MTF false was rather small, however it worked in the opposite direction, making lookups less efficient. This is likely because no word is searched twice, so shifting the links around only makes finding the next words harder.

Overall, MTF is helpful if the data being parsed contains duplicates of the same words. Shifting searched links to the front of their lists is helpful when certain words are really common, however in its worst case scenario (all unique words) it can do more harm than good.

Question 4:

When looking at the number of links traversed in relation to the size of the hash table, it becomes clear that they have an inverse relationship. As the hash table gets smaller, the number of links traversed increases. This makes sense, considering that the more slots there are for the new hash values, the less collisions that occur. After testing banhammer with different sized hash tables, the differences in look up times became very apparent. A hash table of size 10 requires 11,760 lookups, whereas a hash table of size 100 requires 1413. However, there is an upper limit to the efficiency of hash tables. The size of the table can only improve until each word has a unique hash value. In other words, there are no collisions and the average seek length is 1. At this point, adding more slots will not improve efficiency.