An Evaluation of the Hash Function in CMAP

by Bob Thrun

Hashing converts a character string or to an index into an array. This is done to facilitate searching when the number of character strings is much less than the possible number of character strings. The technique used in CMAP is called hashing with chaining. A survey station name is hashed to see which of many chains (or bins) it belongs to. Then comparisons need be made only to names in that particular chain.

One popular hash function is to treat a character string as a large integer, divide by a prime number, and use the remainder of the division as the hash value.

CMAP folds an 8-byte station name to a 4-byte integer and then divides. In the folding process, the second four bytes are shifted by two bits so that transpositions do not hash to the same value; so ABCD1234 does not hash to the same value as 1234ABCD. This is not a major problem with an 8-character name limit, but would be more important with shorter names. The actual hashing process consists of: shift the second four bytes, XOR the two halves of the name, divide by the hash divisor, and then use the remainder of the division as the hash value. There are some possible variations of this hashing scheme. The second half of the name could be shifted by only one bit. The two halves could be folded together by an AND or SUB operation instead of the XOR. This scheme resembles the generation of a Cyclic Redundancy Check (CRC), though I did not realize it at the time I devised it. It was simpler to write the hash function in assembly language than in Fortran. The programs to test the hashing process are fairly small.

CMAP has an array of links with many chains of name indices running thru it. The chains are singly-linked lists. These are like waiting lines where each individual knows who is in front of him. Initially all the chains have zero length with zero in the pointers to the tail of each chain. New stations are added to the tail of each chain. As stations are added to each of the chains, the pointer to the tail is updated. The searching to match a station name starts at the tail of the chain and goes toward the head. For most survey shots, one of the stations is likely to have just been added to the tail of a chain by the previous shot. The other station is usually new and the search involves comparing its name to all the other stations in its chain. If a link pointing to zero is encountered, the search has reached the head of the line and the station is new

The searching process along each chain is a linear search, which is an Order N^2 process. If there are N unique names in the chain, the first name in the chain involves no comparison of names, the second name requires one comparison, the third name requires two comparisons, etc. For N unique names in a chain, (N-1)N/2 total name comparisons will be done. If we have each name getting a different hash value, and no loops or branches in the survey, then each new shot will require only one name comparison. Loop closures involve two old stations, which means the closing shot requires at least two name comparisons. It is theoretically possible to generate a series of station names that all hash to the same value, but very unlikely.

To assess the efficiency of the hashing process, all the FROM and TO station names from real cave surveys were read. The number of name comparison was counted for hash divisors from 10 to 10,000. A typical set of results is shown in Figure 1. The tallest spikes are for hash divisors that are powers of two: 256, 512. 1024, etc. The next tallest spikes are at combinations of powers of two: 384, 768, etc. The good and poor performing hash divisor values were the same for different cave surveys.

To see how the hash function behaves with different cave surveys, we plotted (comparisons per shot) vs. (shots/(hash divisor)). The (shots/(hash divisor)) is the average number of shots in each chain or bin. The number of shots is used, not the number of unique station names. A shot always requires two name searches. There are usually fewer names than shots. So that the results for different caves could be

plotted on the same graph, we ignored the spikes and got the bottom edge of the data plot by treating it as the bottom perimeter of a convex hull. All the curves were generated with hash divisors from 10 to 10,000. The largest surveys had the most comparisons per shot. The results from 12 caves are shown in Figure 2. All the cave surveys had similar station naming conventions. The asymptotic limit for the number of comparisons per shot depends on the ratio of loops to shots. If each new shot links to the previous shot, then there will be many shots where the old station is matched with one comparison. Reducing the number of comparisons reaches a point of diminishing returns when the time spent doing name comparisons becomes smaller than the time spent reading characters and doing the trig and other math on the survey shots.

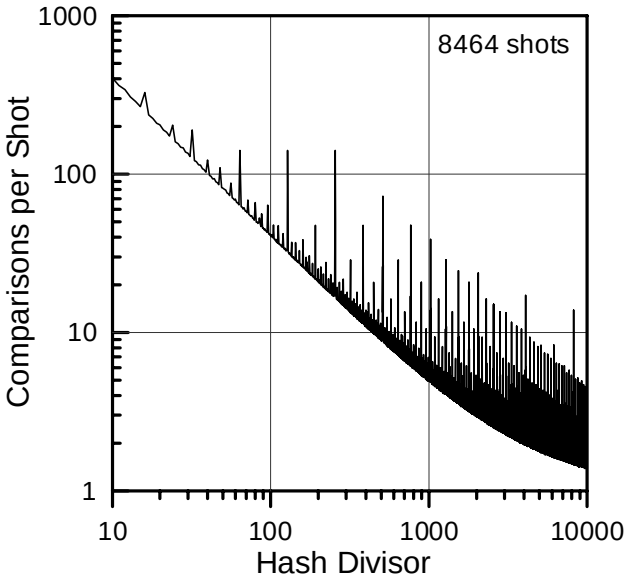

Figure 1. Typical variation of the number of comparisons with the hash divisor.

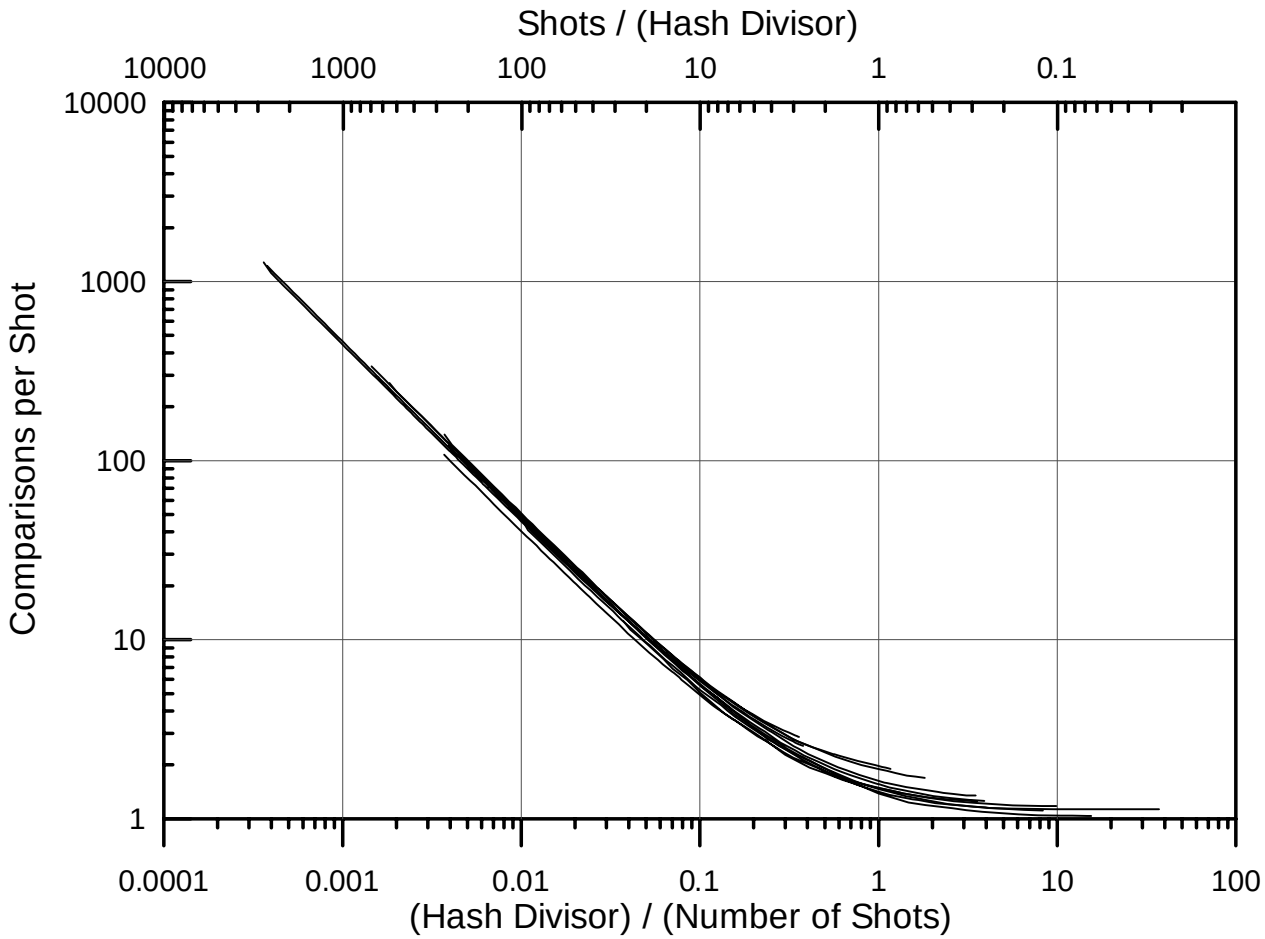

Figure 2. Composite hull bottom for a wide range of survey sizes.