

Double hashing with passbits

Walter A. Burkhard

*Gemini Storage Systems Laboratory, Department of Computer Science and Engineering, University of California,
San Diego, La Jolla, CA 92093-0114, USA*

Received 21 April 2005; received in revised form 1 August 2005; accepted 10 August 2005

Available online 6 September 2005

Communicated by F.Y.L. Chin

Abstract

Double hashing with bucket capacity one is augmented with multiple passbits to obtain significant reduction to unsuccessful search lengths. This improves the analysis of Martini et al. [P.M. Martini, W.A. Burkhard, Double hashing with multiple passbits, *Internat. J. Found. Theoret. Comput. Sci.* 14 (6) (2003) 1165–1188] by providing a closed form expression for the expected unsuccessful search lengths.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Algorithms; Data structures; Analysis of algorithms

1. Introduction

Double hashing is an efficient scheme for the generation of probe sequences within open-addressing. The run-time performance of double hashing is well known via both expected successful and unsuccessful search length expressions [4,7].

Passbits provide a mechanism to improve the unsuccessful search length at the expense of additional space [1,2]. Each bucket has an additional passbit which is initially set false; the passbit is set true, during an insertion, whenever a record overflows beyond the bucket. Subsequent unsuccessful searches will terminate at the first bucket encountered with a false passbit rather than locating an empty bucket.

Multiple passbits utilizes an implicit partitioning of the hash-domain into $g \geq 1$ equal-sized blocks (approximately); each bucket is augmented with g passbits. Initially all passbits are set false. During insertion of record r within block i , passbit i is set true at each full bucket encountered. During an unsuccessful search for block i record \hat{r} , the probe sequence for \hat{r} is followed until a bucket is encountered with passbit i false.

Fig. 1 presents an example with two passbits per bucket; the records are inserted in order: *cat*, *dog*, *sow*, and finally *doe*. Since *doe*, residing within block 1, collides with both *cat* and *dog*, passbit 1 is set at both locations 2 and 1.

The table is configured with $n = 5$ buckets containing $m = 4$ records; the load factor $\alpha = m/n = 0.8$ measures the number of records per bucket. For tables without passbits, the expected unsuccessful search length is $1/(1 - \alpha)$ which

E-mail address: burkhard@cs.ucsd.edu (W.A. Burkhard).

| | | pb_0 | pb_1 | | |
|----|-----|--------|--------|--------------------|-------|
| 0: | doe | F | F | probe sequence | block |
| 1: | cat | F | T | cat: 1, 2, 3, 4, 0 | 0 |
| 2: | dog | F | T | dog: 2, 4, 1, 3, 0 | 0 |
| 3: | | F | F | sow: 4, 2, 0, 3, 1 | 0 |
| 4: | sow | F | F | doe: 2, 1, 0, 4, 3 | 1 |

Fig. 1. Hash table containing four records.

is 5 for Fig. 1 table and the average unsuccessful search length is 3. For tables with $g = 2$ passbits per bucket, the expected unsuccessful search length is $1/(2\sqrt{1-\alpha} - (1-\alpha))$ which is approximately 1.44 for the same table and the average unsuccessful search length is 1.25. The expected unsuccessful search length formula is given within [6]; this formula is generalized here to include additional passbits. Tables configured with $g = 1$ passbit per bucket have been analyzed [3,5]. The expected unsuccessful search length is $1/((1-\alpha)(1-\log(1-\alpha)))$ which is approximately 1.92 for Fig. 1 table. The average unsuccessful search length for the table would be 1.5 with one passbit per bucket.

The average unsuccessful search lengths are calculated by assuming all search configurations are equally-likely. This assumption is typical of general-purpose hashing scheme analysis and is utilized here. Moreover, the utilization of passbits does not modify the successful search length.

2. Passbit result

The unsuccessful search length L_u designates the number of buckets examined to determine a record does not reside within the table. The expected unsuccessful search length result is

Theorem 1. A table utilizing double hashing configured with $g \geq 2$ passbits per bucket with loading factor α has expected unsuccessful search length $E[L_u]$ where

$$E[L_u] = \frac{g-1}{g\sqrt[g]{1-\alpha} - (1-\alpha)}. \tag{1}$$

The $g = 2$ example calculation for Fig. 1 table utilizes Eq. (1).

3. Passbit analysis

A table is configured with n buckets having $g \geq 2$ passbits per bucket. The *construction sequence* for the table is the sequence of indices of buckets successively probed to store the records residing within the table. In the Fig. 1 example, the construction sequence is 1, 2, 4, 2, 1, 0. The length k of the construction sequence divided by the number m of records within the table is the average successful search length for the table. The expected length of the construction sequence is known to be $k = -n \log(1-\alpha)$ [5,6].

The unsuccessful search length L_u designates the number of buckets examined to determine a record r does not reside within the table. Assuming r is within block i , L_u is one plus the number of buckets probed in which passbit i is true. Let \hat{p} designate the probability passbit i is false. Then

$$\text{Prob}(L_u \geq \xi) = (1 - \hat{p})^{\xi-1}, \quad \xi \geq 1$$

and the expected unsuccessful search length is

$$E[L_u] = \sum_{\xi \geq 1} (1 - \hat{p})^{\xi-1} = \frac{1}{\hat{p}}. \tag{2}$$

Symmetry considerations indicate that all passbits have identical expected false probabilities; our calculations are for passbit one. There are two construction sequence configurations demanding passbit one be false.

- (a) The bucket has no block one probes and any number of other probes.
 (b) The bucket has one block one probe and any number of other probes; the block one probe must precede all other probes to this bucket within the construction sequence.

The multinomial probability density P_{j_1, j_2, \dots, j_g} , the probability j_i probes for $1 \leq i \leq g$ of hash-domain block i access a particular bucket within the construction sequence of length k , is

$$P_{j_1, j_2, \dots, j_g} = \binom{k}{j_1, j_2, \dots, j_g} \left(\frac{1}{gn}\right)^{j_1 + j_2 + \dots + j_g} \left(1 - \frac{1}{n}\right)^{k - (j_1 + j_2 + \dots + j_g)}.$$

The multinomial coefficient counts the number of length k sequences

$$\binom{k}{j_1, j_2, \dots, j_g} = \frac{k!}{j_1! j_2! \dots j_g! (k - (j_1 + j_2 + \dots + j_g))!}$$

with j_i block i probes accessing the bucket for $1 \leq i \leq g$.

The probability \hat{p} an unsuccessful search ends at a bucket is the sum of the probabilities p_a for configuration a and p_b for configuration b .

The p_a probability is

$$\begin{aligned} p_a &= \sum_{j_2 + j_3 + \dots + j_g = 0}^k P_{0, j_2, \dots, j_g} \quad \text{the summation includes all } g - 1 \text{ compositions of } 0 \text{ through } k \\ &= \sum_{j_2 + j_3 + \dots + j_g = 0}^k \binom{k}{j_2, j_3, \dots, j_g} \left(\frac{1}{gn}\right)^{j_2 + j_3 + \dots + j_g} \left(1 - \frac{1}{n}\right)^{k - (j_2 + j_3 + \dots + j_g)} \\ &= \sum_{\xi=0}^k \binom{k}{\xi} \left(\frac{g-1}{gn}\right)^\xi \left(1 - \frac{1}{n}\right)^{k-\xi} = \left(1 - \frac{1}{gn}\right)^k \approx e^{-k/gn} \\ &= \sqrt[g]{1 - \alpha}. \end{aligned}$$

Similarly the p_b probability is

$$\begin{aligned} p_b &= \sum_{j_2 + j_3 + \dots + j_g = 0}^{k-1} P_{1, j_2, \dots, j_g} \left(\frac{1}{1 + j_2 + j_3 + \dots + j_g}\right) \quad \text{the summation includes all } g - 1 \\ &= \sum_{j_2 + j_3 + \dots + j_g = 0}^{k-1} \binom{k}{1, j_2, \dots, j_g} \left(\frac{1}{gn}\right)^{1 + j_2 + \dots + j_g} \left(1 - \frac{1}{n}\right)^{k - (1 + j_2 + \dots + j_g)} \left(\frac{1}{1 + j_2 + \dots + j_g}\right) \\ &= \frac{1}{g-1} \sum_{\xi=0}^{k-1} \binom{k}{\xi+1} \left(\frac{g-1}{gn}\right)^{\xi+1} \left(1 - \frac{1}{n}\right)^{k - (\xi+1)} \\ &= \frac{1}{g-1} \left(1 - \frac{1}{gn}\right)^k - \frac{1}{g-1} \left(1 - \frac{1}{n}\right)^k \approx \frac{1}{g-1} (e^{-k/gn} - e^{-k/n}) \\ &= \frac{1}{g-1} (\sqrt[g]{1 - \alpha} - (1 - \alpha)). \end{aligned}$$

Accordingly

$$\hat{p} = p_a + p_b \approx \frac{1}{g-1} (g \sqrt[g]{1 - \alpha} - (1 - \alpha)). \quad (3)$$

This identity was utilized

$$\frac{1}{\binom{q+\xi}{q}} \sum_{j_2 + j_3 + \dots + j_g = \xi} \binom{k}{q, j_2, \dots, j_g} = \binom{k}{\xi + q} (g-1)^\xi$$

as well as the formula $k/n = -\log(1 - \alpha)$.

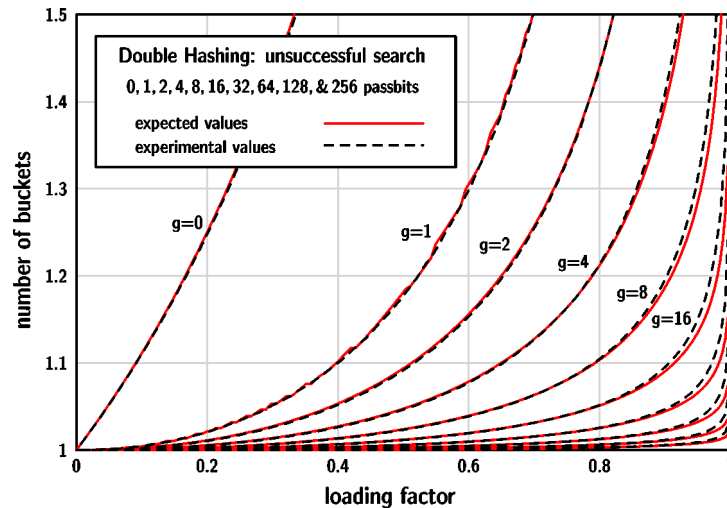


Fig. 2. Unsuccessful search lengths for $g = 0, 1, 2, 4, \dots, 128, 256$ passbits per bucket.

4. Experimental results

Fig. 2 presents experimental data in graphical form together with the expected values calculated via Eq. (1); the table size is 131 buckets. The expected values for either zero or one passbit per bucket are calculated using the well-known formulae [4,5].

5. Implementation

The pseudo-code shows the simplicity of the scheme; the passbit selection `pbit` is independent of both the index and step values.

```
boolean
access ( Table table, Data dat )
{
    unsigned val , index , step , pbit ;

    if ( table == NULL ) return false ;
    val = table->value ( data ) ;
    index = val mod table->size ;
    step = val mod ( table->size - 1 ) + 1 ;
    pbit = ( val / ( table->size * ( table->size - 1 ) ) ) mod table->g ;
    .....
}
```

6. Conclusions

The expected unsuccessful search length for double hashing can be significantly reduced by the use of passbits, often an insignificant appendage of additional space. Our formulae provide accessible performance measures.

This scope of this approach could be greatly enlarged via an accessible formulation for k in situations in which buckets contain a fixed number of records greater than one.

References

- [1] O. Amble, D.E. Knuth, Ordered hash tables, *Comput. J.* 17 (2) (1974) 135–142.
- [2] K. Furukawa, Hash addressing with conflict flag, *Inform. Process. Japan* 13 (1973) 13–18.

- [3] T. Gunji, Analysis of hash addressing methods, Technical report TR-76-03, Department of Information Science, University of Tokyo, Tokyo, Japan, 1976.
- [4] D.E. Knuth, *Sorting and Searching*, second ed., The Art of Computer Programming, vol. 3, Addison-Wesley, Reading, MA, 1998.
- [5] P.-A. Larson, Analysis of uniform hashing, *J. ACM* 30 (4) (1983) 805–819.
- [6] P.M. Martini, W.A. Burkhard, Double hashing with multiple passbits, *Internat. J. Found. Theoret. Comput. Sci.* 14 (6) (2003) 1165–1188.
- [7] W.W. Peterson, Addressing for random-access storage, *IBM J. Res. Develop.* 1 (2) (1957) 130–146.