Notes about d-cuckoo hashing

Jurek Czyzowicz Wojciech Fraczak Feliks Welfeld

May 22, 2006

Abstract

We show an upper bound for the minimal size set of keys which cannot be inserted into a d-cuckoo hashing table independently of the hashing functions and the insertion algorithm. We also discuss the probability for a successful insertion into such a hash table.

1 Introduction

We are considering a hashing table structure which is a combination of d-ary cuckoo hashing and filter hashing, as described in [FPSS03]. Cuckoo hashing was introduced by Pagh and Rodler in [PR01]. It consists of two hash tables, T_1 and T_2 of size r, together with two hashing functions h_1, h_2 into $[0, r-1]$. An entry e can reside at position $h_1(e)$ in T_1 or at positon $h_2(e)$ in T_2. Insertion of an element e ismade according to the following schema. Place e in the position $h_1(e)$ of T_1. If $h(e)$ was occupied by some other element e' then relocate e' to the new location $T_2[h_2(e')]$. If in turn, this position was occupied by another e'' then move e'' to the location $T_1[h_1(e'')]$, and so on, until an empty position is obtained. It is possible that a free position will not be found and the algorithm must proceed to a reconstruction of the hashing tables. D-cuckoo hashing is a generalization of cuckoo hashing and filter hashing. The d-cuckoo hash table is divided into d "layers", d hash functions are used to compute a position of a given element in each layer.Lookup and deletion require at most d probes, one in each layer, so they can be executed in parallel if layers are implemented in different banks of memory. However, in contrast to filter hashing, we do not assume immediate insertion failure when all d positions computed for an element are already occupied, but we

do allow for "bouncing" of keys to new positions, as in the original cuckoo hashing.

It is obvious that such hashing table will accommodate at least d keys even in degenerate case when all d hash functions return the same constant value. One may hope that with "good" hash functions, one can guarantee insertion of the arbitrary set of keys of some size into the table. We demonstrate that with any set of hash functions, there exists some quite small set of keys which cannot be inserted into the table.

2 Notation

Let x, y be two integers such that $x \leq y$. By $[x, y]$ we denote the set of all integers from x to y, x and y inclusive. By $\mathcal{N}$ we denote the set of all natural numbers.

A *d-cuckoo hashing table* is defined by the following:

- M — the set of all possible *keys* (by $|M|$ we denote the size of M)
- d — the number of dimensions ($d \geq 1$)
- $k_1, k_2, \ldots, k_d$ — respective sizes of the dimensions, for $i \in [1, d]$
- $f_1, f_2, \ldots, f_d$ — *hashing functions*, $f_i : M \mapsto [0, k_i - 1]$

3 Description of the problem

We introduce the following definition of an *allocatable set of keys*.

Definition 1. We say that a set of keys $S \subset M$ is *allocatable* within a given d-cuckoo hashing table if there exists an injective mapping $F : S \mapsto [1, d] \times \mathcal{N}$ such that for every $s \in S$, $F(s) = (i, f_i(s))$ for $i \in [1, d]$.

The above definition does not assume any properties of the hashing functions or of the insertion algorithm. The definition states that a set S of keys is allocatable if there exists a representation of S in the d-cuckoo hashing table. Note that a specific insertion algorithm may fail to place in memory an allocatable set of keys.

Problem: What is the minimal size of a non-allocatable set for a given d-cuckoo hashing table?

More precisely, given a d-cuckoo hashing table, what is the smalest possible integer n such that for any hashing functions used, we can always find $S \subseteq M$ such that S is non-allocatable and $|S| \leq n$. For example, not that for $d = 2$ and $k_1 = k_2 = k$ and $|M| > 2k^2$ we can find a small non-allocatable set S (i.e., $|S| = 3$) of keys, because it is then possible to find some 3 keys having the same hashing values for both hashing functions.

Obviously, $d < n \leq \sum_{i=1}^{d} k_i$.

4 Upper bound for minimal size of non-allocatable set

In order to calculate an upper bound for the minimal size of a non-allocatable set, we introduce a notion of *allocation area* of a set $S \subset M$ of keys, denoted by $I(S)$. Intuitively, $I(S)$ defines those elements of the hashing table, $I(S) \subseteq [1, d] \times \mathcal{N}$, which may be used in a representation of S. More formally:

$$I(S) \stackrel{\texttt{def}}{=} \{(i, f_i(s)) \mid s \in S\}.$$

By definition, every key s is mapped through $f_1, f_2, \ldots, f_d$ into d different pairs $(i, f_i(s))$, for $i \in [1, d]$. Thus, for the singleton set of keys, $S = \{s\}$, the allocation area consists of d elements: $I(S) = \{(1, f_1(s)), (2, f_2(s)), \ldots, (d, f_d(s))\}$.

Proposition 1. *Let $S \subset M$ be a set of keys. If the size of S is strictly bigger than the size of $I(S)$, then S is non-allocatable.*

Proof. By pidgeon principle, a representation of S has to use at least as many elements of the hashing table as the size of S. □

Corollary 2. *Let $S \subset M$ be a set of keys such that the size of S is strictly bigger than the size of $I(S)$. Every subset S' of S of size bigger than $|I(S)|$, i.e., $|S'| > |I(S)|$, is non-allocatable.*

In order to calculate an upper bound for a minimal non-allocatable set S we will consider its possible allocation areas. We have already seen that a singleton set $\{s_1\}$ has the allocation area of size d. If there are more than d different keys, $s_1, s_2, \ldots, s_d, s_{d+1}, \ldots$, which all have the same allocation area,

then, by Corollary 2, the set $\{s_1, s_2, \ldots, s_d, s_{d+1}\}$ is non-allocatable. Such a situation may occur whenever $|M| > \prod_{i=1}^{d} k_i$, or when the hashing functions $f_1, f_2, \ldots, f_d$ are not uniform (i.e., are not good hashing functions).

Definition 2. Let $I(S)$ be an allocation area of a set S of keys. We call *profile* of $I(S)$ the vector of d integers $(x_1, x_2, \ldots, x_d)$ such that $x_i \stackrel{\texttt{def}}{=} |\{(i, f_i(s)) \mid s \in S\}|$.

Intuitively, profile $(x_1, x_2, \ldots, x_d)$ corresponds to an allocation area with exactly x_1 elements from dimension 1, x_2 elements from dimension 2, etc. An allocation area of profile $(x_1, x_2, \ldots, x_d)$ has exactly $\sum_{i=1}^{d} x_i$ elements. A lower bound for the maximum size set S with a corresponding allocation area (i.e., with exactly x_1 elements from dimension 1, x_2 elements from dimension 2, etc.) can be calculated.

Lemma 3. *For every profile $(x_1, x_2, \ldots, x_d)$ there exists a set $S \subset M$ with the allocation area of profile $(x'_1, x'_2, \ldots, x'_d)$ with $x'_i \leq x_i$, for $i \in [1, d]$, and such that*

$$|S| \geq |M| \prod_{i=1}^{d} \frac{x_i}{k_i}$$

Proof. By construction of such S:

1 $S := M$ --- i.e., $|S| \geq |M|$
2 **for** $i = 1$ **to** d **do**
3 Choose a set $Y = \{y_1, \ldots, y_{x_i}\}$ of x_i different integers such that $f_i^{-1}(Y) \geq |S| k_i / x_i$, which is always possible.
4 $S := f_i^{-1}(Y)$ --- i.e., $|S| \geq |M| \prod_{j=1}^{i} k_j / x_j$
5 **end for**

□

Corollary 2 and Lemma 3 directly implythe following.

Theorem 4. *The existence of a profile $(x_1, \ldots, x_d)$ such that:*

$$|M| \prod_{i=1}^{d} \frac{x_i}{k_i} > \sum_{i=1}^{d} x_i \tag{1}$$

implies the existence of a non-allocatable set S of $(\sum_{i=1}^{d} x_i + 1)$ elements.

Inequality 1 can be rewritten into:

$$\prod_{i=1}^{d} x_i > \left(\frac{\prod_{i=1}^{d} k_i}{|M|} \right) \sum_{i=1}^{d} x_i$$

Lemma 5. *Let (x, y) be two positive integers such that $x > y + 1$. We have:*

$$xy < (x-1)(y+1)$$

Lemma 5 extends directly to any vector of positive integers. Thus, in order to find a profile verifying Inequality 1 and minimizing the sum of its elements, we will choose all integers of the profile being equal, i.e., $(x, x, \ldots, x)$, with $x \leq k_i$, for $i \in [1, d]$. In this case Inequality 1 can be rewritten into:

$$x^d > \left(\frac{\prod_{i=1}^{d} k_i}{|M|} d \right) x$$

Whenever $d > 1$, the condition

$$x > \left(\frac{\prod_{i=1}^{d} k_i}{|M|} d \right)^{\frac{1}{d-1}}$$

implies Inequality 1.

We denote by $B(|M|, d, k_1, \ldots, k_d)$ the size of a smallest non-allocatable subset of M. If $k_1 = k_2 = \ldots = k_d = k$ then we will write $B(|M|, d, k)$ in order to denote $B(|M|, d, k_1, \ldots, k_d)$.

Theorem 6.

$$B(|M|, d, k_1, \ldots, k_d) \leq d \left\lfloor \left(\frac{\prod_{i=1}^{d} k_i}{|M|} d \right)^{\frac{1}{d-1}} + 1 \right\rfloor \tag{2}$$

Corollary 7. *In the case of a uniform layer d-cuckoo hashing, i.e., when $k_1 = k_2 = \ldots = k_d = k$, we have:*

$$B(|M|, d, k) \leq kd \left(\sqrt[d-1]{\frac{kd}{|M|}} - 1 \right). \tag{3}$$

5 Estimating probability for a successful insertion in d-ary cuckoo hashing

The d-ary cuckoo hashing table is defined by m, d, and k, where m is the size of the key space, d is the number of dimensions, k is the size of one dimension (we assume that all dimensions are of the same size). Thus, the capacity of the d-ary cuckoo hashing table is dk.

In this section we will consider the probablility that d-ary cuckoo insertion succeeds. We cocentrate on the simpler case of a uniform layer d-cuckoo hashing, i.e., having all dimensions of the same size k. The capacity of the d-ary cuckoo hashing is dk.

By r we denote the filling ratio of the d-cuckoo hashing table, i.e., $r = \frac{n}{dk}$, where n is number of elements stored in the table.

The size of the minimal non-allocatable set of keys was expressed by formula $B(m, d, k)$ (Inequation 7) for an upper bound. Due to the Hall's Theorem (see, e.g., [CLR90]), the formula for the upper bound can be considered as the exact (with accuracy $\pm d$) bound of the size of the minimal non-allocatable set. Thus, every set smaller than $B(m, d, k) - d$ can always be inserted (whenever hashing functions are uniform over the universe).

5.1 Assumptions

We suppose that all hashing functions $f_1, \ldots, f_d$ are "good", i.e., the probablity of $f_i(x) = y$ is the same, for any key x, $i \in [1, d]$, and $y \in [1, k]$, and all those probabilities are independent. Therefore, the probability that $f_i(x)$ indexes a "busy" element of the d-cuckoo hashing table is r, denoted by $P(f_i(x) \text{ is busy}) = r$.

5.2 Estimates

5.2.1 Direct insertion (no bumping)

The probability of the hashing conflict on all d dimensions for an element is r^d. Thus, the probability of a direct insertion (without moving other elements) of the element into the d-ary cuckoo hashing table is $1 - r^d$.

5.2.2 Single bump insertion

The probability of impossibility of insertion with at most one bump (i.e., none or one element which is already present in the table is moved) is $r^d r^{d(d-1)} = r^{d^2}$, i.e., there is a hashing conflict on all d dimensions for x and there is a hashing conflict on $d-1$ dimensions for every element y_i conflicting with x on dimension i.

5.2.3 Unlimited bumping insertion

A perfect matching in a bipartite graph can be represented by a digraph, see [FPSS03]. In our case, the bipartite graph is build out of the set S of keys stored in the d-cuckoo hash table and the allocation area $I(S)$ with the edges defined by hashing functions, $(x, f_i(x))$, i.e., $G = (S \cup I(S), \{(x, (i, f_i(x))) \mid x \in S, i \in [1, d]\})$. A perfect matching of such a graph means that S can be fully represented within the d-cuckoo hashing table. In such a case, a particular perfect matching is represented by digraph $\vec{G}$ by orienting the edges of G in the following way: an edge $(x, (i, f_i(x)))$ is oriented from $(i, f_i(x))$ towards x if and only if x is stored in dimension i at location $f_i(x)$. Otherwise, the edge $(x, (i, f_i(x)))$ is oriented from x towards $(i, f_i(x))$ (see Figure 1).

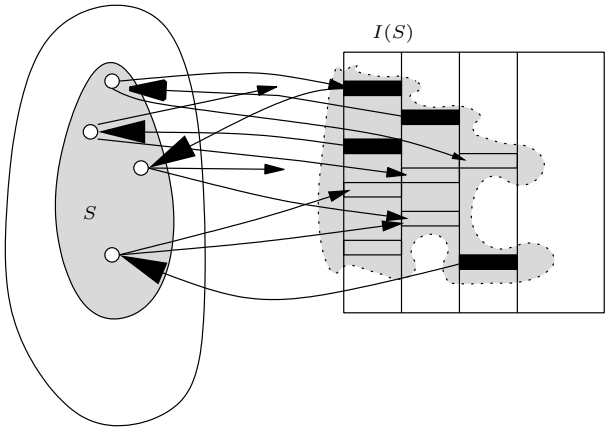

Figure 1: Perfect matching represented by a bipartite digraph

An element $x \notin S$ can be successfully inserted into the d-cuckoo hashing table if and only if:

- x can be directly inserted (i.e., there exists a dimension $i \in [1, d]$ for which $f_i(x)$ is not busy); or

- there exists a directed path in $\vec{G}$ starting in $f_i(x)$, for some $i \in [1, d]$ and ending in a non-busy location of the d-cuckoo hash table.

Let N be the number of edges of $\vec{G}$ directed from S towards $I(S)$, which are reachable from $f_1(x)$, $f_2(x)$, ..., or $f_d(x)$. Assuming full randomness of destinations of those edges, the probability that everyone is pointing into a busy location is r^N.

The value of N is smaller than $|S|(d-1) = rkd(d-1)$ and bigger than $B(m, d, k)(d-1)$. Thus, if $rkd \geq B(m, d, k)$, then the probability that an element cannot be inserted is at most $r^{B(m,d,k)(d-1)}$. Therefore, the probability of a single successful insertion is:

$$P(\text{successful insertion}) \geq 1 - r^{B(m,d,k)(d-1)} \ . \tag{4}$$

If $\frac{N}{d-1} < B(m, d, k) - d$, i.e., the number of elements which are reachable from the inserting element x is smaller than the size of the minimal non-allocatable set, the probability of a successful insertion of x is 1.

6 Conclusion

We proved that for a d-cuckoo hashing table defined by $|M|, d, (k_1, \ldots, k_d)$ (see Section 2) there always exists a set S of size $B(|M|, d, k_1, \ldots, k_d)$:

$$B(|M|, d, k_1, \ldots, k_d) \stackrel{\texttt{def}}{=} d \left\lfloor \left(\frac{\prod_{i=1}^{d} k_i}{|M|} d \right)^{\frac{1}{d-1}} + 1 \right\rfloor \tag{5}$$

which cannot be insterted into the hashing table. E.g., for $|M| = 2^{144}$, $d = 16$, and $k_1 = k_2 = \ldots = k_{16} = 2^{12}$, there exists a non-allocatable set of size 192.

It is important to underline that this in an upper bound which is independent on the hashing function used.

On the other hand, the probability of insertion failure is very low if uniform distribution is assumed.

References

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[FPSS03] Dimitris Fotakis, Rasmus Pagh, Peter Sanders, and Paul G. Spirakis. Space efficient hash tables with worst case constant access time. In *STACS 2003*, volume 2607 of *LNCS*, pages 271–282. 2003.

[FW04] Wojciech Fraczak and Feliks Welfeld. What's the minimal size of a set of keys which cannot be inserted into a d-cuckoo hashing table? Technical report, IDT Canada, 1575, Carling Avenue, Ottawa, Ontario, K1Z 7M3 Canada, May 2004.

[PR01] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. In *BRICS Report Series*, volume RS-01-32. Department of Computer Science, University of Aarhus, 2001.