A CUCKOO HASHING VARIANT WITH IMPROVED MEMORY UTILIZATION AND INSERTION TIME

ELY PORAT AND BAR SHALEM

Department of Computer Science Bar Ilan University, Ramat Gan, 52900, Israel

1. ABSTRACT

Cuckoo hashing [4] is a multiple choice hashing scheme in which each item can be placed in multiple locations, and collisions are resolved by moving items to their alternative locations. In the classical implementation of two-way cuckoo hashing, the memory is partitioned into contiguous disjoint fixed-size buckets. Each item is hashed to two buckets, and may be stored in any of the positions within those buckets. Ref. [2] analyzed a variation in which the buckets are contiguous and overlap. However, many systems retrieve data from secondary storage in same-size blocks called pages. Fetching a page is a relatively expensive process; but once a page is fetched, its contents can be accessed orders of magnitude faster. We utilize this property of memory retrieval, presenting a variant of cuckoo hashing incorporating the following constraint: each bucket must be fully contained in a single page, but buckets are not necessarily contiguous. Empirical results show that this modification increases memory utilization and decreases the number of iterations required to insert an item. If each item is hashed to two buckets of capacity two, the page size is 8, and each bucket is fully contained in a single page, the memory utilization equals 89.71% in the classical contiguous disjoint bucket variant, 93.78% in the contiguous overlapping bucket variant, and increases to 97.46% in our new non-contiguous bucket variant. When the memory utilization is 92% and we use breadth first search to look for a vacant position, the number of iterations required to insert a new item is dramatically reduced from 545 in the contiguous overlapping buckets variant to 52 in our new non-contiguous bucket variant. In addition to the empirical results, we present a theoretical lower bound on the memory utilization of our variation as a function of the page size.

2. Introduction

Cuckoo hashing [4] is a multiple choice hashing scheme in which each item can be placed in multiple locations, and collisions are resolved by moving items to their alternative locations. This hashing scheme resembles the cuckoo's nesting habits: the cuckoo lays its eggs in other birds' nests. When the cuckoo chick hatches, it pushes the other eggs out of the nest. Hence the name "cuckoo hashing." As Ref. [2] explains, analysis of hashing is similar to the analysis of balls and bins. Hashing an item to a memory location corresponds to throwing a ball into a bin. Insights from balls and bins processes led to breakthroughs in hashing methods. For example, if we throw n balls into n bins independently and uniformly, it is highly probable that the largest bin will get $(1+\mathrm{o}(1))\log(n)/\mathrm{loglog}(n)$ balls. Azar et. al [10] found that if

1068-0314/12 $26.00 © 2012 IEEE
DOI 10.1109/DCC.2012.41

each ball selects two bins independently and uniformly, and is placed in the bin with fewer balls, the final distribution is much more uniform. This led to hashing each item to one of two possible buckets, decreasing the load on the most-loaded bucket to $\log(\log(n)) + \mathrm{O}(1)$ with high probability. In general, if each item is hashed into $d \geq 2$ buckets, the maximum load decreases to $\log(\log(n))/\log(d) + \mathrm{O}(1)$.

Cuckoo hashing [4] is an extension of two-way hashing. Each item is hashed to a few possible buckets, and existing items may be moved to their alternate buckets in order to free space for a new item. There are many variants of cuckoo hashing. The goals of cuckoo hashing are to increase memory utilization (the number of items that can be successfully hashed to a given memory size) and to decrease insertion complexity. Pagh and Rodler [4] analyzed hashing of each item to $d = 2$ buckets of capacity $k = 1$, and demonstrated that moving items during inserts results in 50% space utilization with high probability. Fotakis et. al. [5] analyzed hashing of each item into more than two buckets. Ref. [6] analyzed a practically-important case in which each item is hashed to $d = 2$ buckets of capacity $k = 2$. Refs. [7, 8] found tight memory utilization thresholds for $d = 2$ buckets of any size $k \geq 2$. Specifically, they proved that the memory utilization for $d = 2$ and $k = 2$ is 89.7%.

Ref. [1] proved that the maximum memory utilization thresholds for $d \geq 3$ and $k = 1$ are equal to the previously known thresholds for the random k-XORSAT problem. Ref. [12, 13] developed a tight formula for memory utilization for any $d \geq 3$ and $k = 1$ and Ref. [11] extended the formula to any $d \geq 3$ and $k \geq 1$.

comment added. While this work was being completed, we became aware of Ref. [14] which proposed a model where the memory is divided into pages and each key has several possible locations on a single page as well as additional choices on a second backup page. They provide interesting experimental results.

In a classical implementation of two-way cuckoo hashing, the memory is partitioned into contiguous disjoint fixed-sized buckets of size k . Each item is hashed to 2 buckets and may be stored in any of the $2k$ locations within those buckets. Ref. [2] analyze a variation in which the buckets overlap. For example, if the bucket capacity k is 3, the disjoint bucket memory locations are: $\{0,1,2\},\{3,4,5\},\{6,7,8\},\ldots$. whereas the overlapping bucket memory locations are: $\{0,1,2\},\{1,2,3\},\{3,4,5\},\ldots$.

Their empirical results show that this variation increases memory utilization from 89.7% to 96.5% for $d = 2$ and $k = 2$. However, many systems retrieve data from secondary storage in same-size blocks called pages. Fetching a page is a relatively expensive process, but once a page is fetched, its contents can be accessed orders of magnitude more quickly. We utilize this property of memory retrieval to present a variant of cuckoo hashing requiring that each bucket be fully contained in a single page but buckets are not necessarily contiguous.

In this paper we compare the following three variants of cuckoo hashing:

(1) CUCKOO-CHOOSE-K- the algorithm introduced in this paper. The buckets are *any* k cells in a page, not necessarily in contiguous locations. There are $\binom{t}{k}$ buckets in a page, where t is the size of the page.

(2) CUCKOO-OVERLAP [2]- The buckets are contiguous and overlap. Here we assume that all buckets are fully contained in a single page, so there are $t-k+1$ buckets in a page. This is a generalization of Ref. [2]. Originally Ref. [2] did not consider dividing the memory into pages.

(3) CUCKOO-DISJOINT [4]- The buckets are contiguous and not overlapping. There are t/k buckets in a page. This is a generalization of Ref. [4]. Originally Ref. [4] did not consider larger buckets.

Note that algorithm CUCKOO-DISJOINT is the extreme case of the CUCKOO-OVERLAP and CUCKOO-CHOOSE_K algorithms when the size of the page t equals the size of the bucket k.

We prove theoretically and present empirical evidence that our CUCKOO-CHOOSE-K modification increases memory utilization. Moreover, using the classical cuckoo hashing scheme, an item insertion requires multiple look-ups of candidates to displace. Empirical results show that our modification dramatically decreases the number of candidate look-ups required to insert an item compared to Ref. [2]. In the overlapping buckets variant [2], some buckets are split between two pages, so that each item resides in up to $2d$ pages. In our variant, each bucket is fully contained in a single page.

An appealing experimental result is that CUCKOO-CHOOSE-K memory utilization converges very quickly as a function of the page size t. When $k = 2$ and $t = 16$, memory utilization is 0.9763. This value is almost identical to memory utilization when $t = 2^{20}$ which equals 0.9767. CUCKOO-OVERLAP memory utilization when $t = 16$ is 0.9494, and CUCKOO-DISJOINT memory utilization is only 0.8970. If we allow a tiny gap of one inside the buckets ($t = 3$), memory utilization increases from 0.9229 (CUCKOO-OVERLAP) to 0.9480 (CUCKOO-CHOOSE-K). Table 4.2 specifies the parameters used in our analysis.

3. Theoretical analysis

We can determine the success of cuckoo hashing by analyzing the cuckoo hyper graph. The vertices of the graph are the memory locations. The hyper-edges of the graph connect all the memory locations where each item could be placed. For example, If we are given 3 items where each item can be placed in memory location 0 or 1, then our graph has 3 edges that connect vertices 0 and 1. The third insertion will fail since we cannot place 3 items in 2 memory locations no matter who we decide to evict in the insertion process. In general, it is well known (see, e.g., ref. [2] for a proof) that a cuckoo hash *fails* if and only if there is a sub-graph S with v vertices and more than v edges. We say that a sub-graph S has *failed* if it has more edges than vertices.

We will begin by analyzing the probability of success of CUCKOO-CHOOSE-K for the case where the page size t equals the array size n. This simple and special case is presented here to introduce the main ideas applied in the following section, where we analyze the general case where the page size is a finite constant (independent of n). Recall that each item can be placed in d buckets chosen uniformly and independently of other items and each bucket is composed of any k locations in a page.

3.1. Memory utilization when the page size t equals the array size n

. An analysis of memory utilization has been performed previously in [5]. In their analysis, they assume $k = 1$ and prove that if $d \geq \frac{2}{\beta} \log\left(\frac{e\beta}{1-\beta}\right)$, then the hashing will be successful with a probability of at least $1 - \mathrm{O}(n^{4-2d})$. Here we derive a similar constraint on memory utilization β. We solve the constraint numerically for different

Symbol	Description	Comments
n	number of vertices (hash table capacity)	$n \to \infty$
m	number of edges (hashed items)	$m \leq n$, $m \to \infty$
d	number of buckets each item is hashed to	typical value: 2, $dk > 2$
k	bucket size	typical value: 2-3, $dk > 2$
t	page size	$t \geq k$. k divides t. t divides n.
g	number of pages	$g = n/t$.
β	memory utilization	$\beta = \frac{m}{n}$,$0 < \beta \leq 1$
V	a set of vertices	$\|V\| = v$
$S = S$(V,E)	a sub-graph	
v	$v = \|V\|$	$dk \leq v < m$. (if $v \geq m$ or $v < dk$ then S cannot fail).
x	$x = \frac{v}{n}$	$\frac{dk}{n} \leq x < \beta$.
ϵ	a small constant	$0 < \epsilon \ll 1$
δ	a small constant	$0 < \delta \ll 1$

TABLE 1. Parameter names and descriptions

values of k and d, and obtain a lower bound on possible memory utilization for the specified values. We perform the analysis using a modification of the method in [5], which we will later generalize to page sizes being equal to any given constant.

We will bound the failure probability using the union bound. But first, we would like to reduce redundant summations. We observe that:

- If there exists a sub-graph $S(V, E)$ with $|E| > |V|$ and there exists an edge that has exactly one vertex v_0 outside of V, then the sub-graph $S'(V' = V \cup \{v_0\}, E')$ also has more edges than vertices since $|E'| \geq |E| + 1 > |V| + 1 = |V'|$.
- If there exists a sub-graph $S(V, E)$, such that $|V| = v$ and $|E| > v + 1$ then there exists a sub-graph $S'(V', E')$ such that $|E'| = |V'| + 1$. We can find such a sub-graph simply by adding vertices to V one by one until we get a sub-graph where the number of edges equals the number of vertices plus one.

For each sub-graph S we define an indicator variable Z_S.

$$Z_S = \begin{cases} 1 & S(V,E) \text{ has } |V| = v \text{ vertices and } |E| = v+1 \text{ edges AND} \\ & \text{There is no edge that connects } V \\ & \text{to exactly one vertex from outside of } V \\ 0 & \text{otherwise} \end{cases}$$

If $Z_s = 1$, then we will say that S is a *bad* sub-graph, and otherwise we will say that S is a *good* sub-graph. If the sum over Z_s of all sub-graphs is equal to zero then every sub-graph is good then the cuckoo hash *succeeded*. We will find the the memory utilization β such that the sum over Z_s of all sub-graphs is o(1) as $n \to \infty$.

Let p_{hit} be the probability that a random edge hits V. Let p_1 be the probability that a random edge connects V to exactly one vertex from outside of V and let $p_{bad}(v)$ be probability that *a given* sub-graph $S(V, E)$ is bad.

Lemma 1. $p_{bad}(v) = \binom{m}{v+1} p_{hit}^{v+1} (1 - p_1 - p_{hit})^{m-(v+1)}$

Proof. Immediate from the definition of Z_s. For S to be bad, exactly $v+1$ edges out of m edges must hit V and all the rest must miss V and must not connect V to exactly one vertex from outside of V. □

Lemma 2. *The probability that a random edge hits V is* $p_{hit}(v) = \left(\frac{\binom{v}{k}}{\binom{n}{k}} \right)^d$

Proof. Each item is hashed independently to d buckets and the size of each bucket is k. The number of buckets in V is therefore $\binom{v}{k}$ and the total number of buckets is $\binom{n}{k}$. □

Lemma 3. *The probability that a random edge connects V to exactly one vertex from outside of V is* $p_1 = d \frac{(n-v)\binom{v}{k-1}}{\binom{n}{k}} \left(\frac{\binom{v}{k}}{\binom{n}{k}} \right)^{d-1}$.

Proof. $d-1$ buckets must all fall in V and one bucket must contain any $k-1$ vertices from V and any of the $n-v$ vertices from outside of V. □

Let $P_{bad}(v)$ be the probability that there exists a sub-graph S with v vertices such that S is bad. According to the union bound, $P_{bad}(v) < N(v) \cdot p_{bad}(v)$, where $N_v = \binom{n}{v}$ is the number of sub-graphs with v vertices. We are going to analyze $P_{bad}(v)$ as $n \to \infty$. If for all v, $P_{bad}(v) = \mathrm{o}\left(\frac{1}{n}\right)$, then $\sum_{v=dk}^{n} P_{bad}(v) \leq \mathrm{o}(1)$ and the cuckoo hash succeeds with high probability. The analysis is similar to the analysis given in [5]. Let $x_0 = \exp\left(\frac{-2}{dk-2}\right)$. We divide the analysis into two sections. In section 3.1.1 we show that for any memory utilization β and $\forall \frac{dk}{n} \leq x < x_0$, $P_{bad}(x)$ is $\mathrm{o}\left(\frac{1}{n}\right)$. In section 3.1.2we find the maximum memory utilization β such that $\forall x_0 \leq x < \beta$, $P_{bad}(x)$ is exponentially small.

3.1.1. $P_{bad}(x)$ *Analysis for* $\frac{dk}{n} \leq x < x_0$. In this section we show that if $dk > 2$ then for any memory utilization β and $\forall \frac{dk}{n} \leq x < x_0$, $P_{bad}(x)$ is $\mathrm{o}\left(\frac{1}{n}\right)$.

Theorem 4. *Let δ be a small constant, $0 < \delta \ll 1$. If $dk \geq 3$ then for any load β:*

1. *If $\frac{dk}{n} \leq x \leq \frac{dk}{n} + \delta$, then $P_{bad}(x) < O\left(n^{-\left((dk)^2 - dk - 1\right)}\right) = o\left(\frac{1}{n}\right)$.*
2. *If $\frac{dk}{n} + \delta \leq x \leq x_0 - \delta$, then $P_{bad}(x)$ decreases exponentially as $n \to \infty$.*

Proof. Details in the full version. □

3.1.2. $P_{bad}(x)$ *Analysis for* $x_0 \leq x < \beta$. In this section we are going to find the maximum memory utilization β such that $\forall x_0 \leq x < \beta$, the probability that there exists a bad sub-graph is exponentially small.

Theorem 5. *Let*

$c_5(x, \beta) = \left(\frac{1}{1-x}\right)^{(1-x)} \left(\frac{1}{x}\right)^{x} \left(\frac{\beta}{\beta - x}\right)^{(\beta - x)} \left(\frac{\beta}{x}\right)^{x} x^{dkx} \left(1 - dk(1-x) x^{dk-1} - x^{dk}\right)^{\beta - x}$. *If $c_5(x, \beta) < 1 - \epsilon$ for all $x_0 \leq x < \beta$, then $P_{bad}(x)$ decreases exponentially as $n \to \infty$. Any memory utilization β that satisfies the constraint is a lower bound on the possible memory utilization.*

Proof. Details in the full version. □

Numerical solutions to the constraint $c_5(x,\beta) < 1$ indicate that the memory utilization of the CUCKOO-K algorithm is $\beta_{Choose-2}(k=2,d=2,t=n) > 0.937$ and $\beta_{Choose-3}(k=3,d=2,t=n) > 0.993$. Our empirical results show that $\beta_{Choose-2}(k=2,d=2,t=n) = 0.9768$ and $\beta_{Choose-3}(k=3,d=2,t=n) = 0.9974$. The memory utilization for $kd > 6$ rapidly approaches one. Theoretical analysis performed by [8, 7] provided tight thresholds of the memory utilization of the CUCKOO-DISJOINT algorithm. $\beta_{Disjoint}(k=2,d=2,t=n) = 0.8970$ and $\beta_{Disjoint}(k=3,d=2,t=n) = 0.9592$. Ref. [2] do not provide a theoretical memory utilization threshold for small k. The empirical results of Ref. [2] show that in the CUCKOO-OVERLAP algorithm $\beta_{Overlap}(k=2,d=2,t=n) = 0.9650$ and $\beta_{Overlap}(k=3,d=2,t=n) = 0.9945$. The theoretical analysis of the CUCKOO-DISJOINT algorithm performed in [12, 1, 13] does not apply for $k > 1$ and the theoretical analysis of the CUCKOO-DISJOINT algorithm performed by [11] does not apply for $d < 3$.

3.2. **Memory utilization when the page size t is a given constant .** In this section we analyze the probability that the hashing fails for the case where the page size t equals a constant. Let $P_{fail}(v)$ be the probability that there exists a sub-graph S with v vertices such that S has more edges than vertices and every vertex is in at least one edge. Let $x_1 = \exp\left(\frac{-(k+1)}{dk-(k+1)}\right)$. Here again we divide the analysis into two sections. In section 3.2.1 we show that for any memory utilization β and $\forall \frac{dk}{n} \le x < x_1$, $P_{fail}(x)$ is $o\left(\frac{1}{n}\right)$. In section 3.2.2 we find the maximum memory utilization β such that $\forall x_1 \le x < \beta$, $P_{bad}(x)$ is exponentially small.

3.2.1. *$P_{fail}(x)$ Analysis for $\frac{dk}{n} \le x < x_1$.* In this section we show that for any memory utilization β and $\forall \frac{dk}{n} \le x < x_1$, $P_{fail}(x)$ is $o\left(\frac{1}{n}\right)$.

The analysis here is similar to the case above where $t = n$, however now we need to take into consideration the distribution of the vertices over the pages.

Let V be a given set of vertices, and let v_i be the number of vertices in page i. The probability that a random edge hits V is equal to

$p_{hit}(V,t) = \left(\sum_{i=1}^{g} \frac{\binom{v_i}{k}}{g\cdot\binom{t}{k}}\right)^d \le \left(\frac{1}{g}\sum_{i=1}^{g}\left(\frac{v_i}{t}\right)^k\right)^d$. Let $\widetilde{p}_{hit}(V,t) = \left(\frac{1}{g}\sum_{i=1}^{g}\left(\frac{v_i}{t}\right)^k\right)^d$ be an upper bound on $p_{hit}(V,t)$.

We are going to use the following lemma which states that increasing the page size reduces $\widetilde{p}_{hit}$.

Lemma 6. *For any set of vertices V and any integer c,$\widetilde{p}_{hit}(V,t) \ge \widetilde{p}_{hit}(V,ct)$.*

Proof. Since the function $f(v) = \left(\frac{v}{t}\right)^k$ is convex, for any sequence of c pages, $\frac{1}{c}\left(\left(\frac{v_1}{t}\right)^k + \ldots + \left(\frac{v_c}{t}\right)^k\right) \ge \left(\frac{v_1+\ldots+v_c}{ct}\right)^k$, thus multiplying the page size by a factor c does not increase $\widetilde{p}_{hit}$. For convenience, we restrict our analysis to pages of size $t = k \cdot c$, where c is any integer. The worst case is obtained when $t = k$ which is equivalent to the classical CUCKOO-DISJOINT hashing.

We will now analyze $P_{fail}(x)$ and $P_{bad}(x)$ as $n \to \infty$. Recall that $\beta = \frac{m}{n}$ is the memory utilization, $0 < \beta \le 1$ and $x = \frac{v}{n}$. □

Theorem 7. *Let δ be a small constant, $0 < \delta \ll 1$. If $(d-1)\,dk \geq 3$, then for any load β:*

1. *If $\frac{dk}{n} \leq x \leq \frac{dk}{n} + \delta$, then $P_{fail}(x) < O\left(n^{-((d-1)dk-1)}\right) = o\left(\frac{1}{n}\right)$.*
2. *If $\frac{dk}{n} + \delta \leq x \leq x_1 - \delta$, then $P_{fail}(x)$ decreases exponentially as $n \to \infty$.*

Proof. Details in the full version. □

3.2.2. $P_{bad}(x)$ *Analysis for* $x_1 \leq x < \beta$. In this section we find the maximum memory utilization β such that $\forall x_1 < x < \beta$, the probability that there exists a bad sub-graph is exponentially small. We examine the set of sub-graphs that have a given distribution $\underline{a}$ of vertices over the pages. $\underline{a} = (a_0, ..., a_t)$, where a_i is the number of pages that have i vertices. For example, when the page size t was equal to n and the number of pages g was 1, the number of sub-graphs with v vertices was $\binom{n}{v}$ and the corresponding $\underline{a}$ of those sub-graphs was $\{0, ..., 0, a_v = 1, 0, ..., 0\}$. by definition of $\underline{a}$,

$$\sum_{i=0}^{t} a_i = g \tag{3.1}$$

Lemma 8. *When the page size t is a given constant, the number of different possible values of $\underline{a}$ is polynomial in n.*

Proof. We denote by $\#\underline{a}$ The number of different possible values of $\underline{a}$.
$\#\underline{a} < g^t = \left(\frac{n}{t}\right)^t$. Since t is constant, $\#a$ is polynomial in n. □

Let $P_{bad}(\underline{a})$ be the probability that there exists a bad sub-graph with distribution $\underline{a}$ of vertices over the pages. If $P_{bad}(\underline{a})$ is exponentially small for every $\underline{a}$, then the union bound over a polynomial number of all possible values of a is also exponentially small. Let $p_{bad}(\underline{a})$ be the probability that *a given* sub-graph $S(V, E)$ with $\underline{a} = (a_0, ..., a_t)$ is bad.

Let $\hat{a} = \frac{a}{g}$ be a unit vector. When the page size t is a constant, the probability that a random edge hits V is:

$$p_{hit}(\underline{a}) = \left(\frac{\sum_{i=k}^{t} a_i \binom{i}{k}}{g\binom{t}{k}}\right)^d = \left(\frac{\sum_{i=k}^{t} \hat{a}_i \binom{i}{k}}{\binom{t}{k}}\right)^d \tag{3.2}$$

and the probability that a random edge connects V to exactly one vertex from outside of V is

$$p_1(\underline{a}) = d\left(\frac{\sum_{i=k}^{t} a_i (t-i)\binom{i}{k-1}}{g\binom{t}{k}}\right)\left(\frac{\sum_{i=k}^{t} a_i \binom{i}{k}}{g\binom{t}{k}}\right)^{d-1} \tag{3.3}$$

Using the union bound we obtain that

$$P_{bad}(\underline{a}) < N(\underline{a}) \cdot p_{bad}(\underline{a}) \tag{3.4}$$

where $N(\underline{a})$ is the number of sub-graphs with $\underline{a} = (a_0, ..., a_t)$.

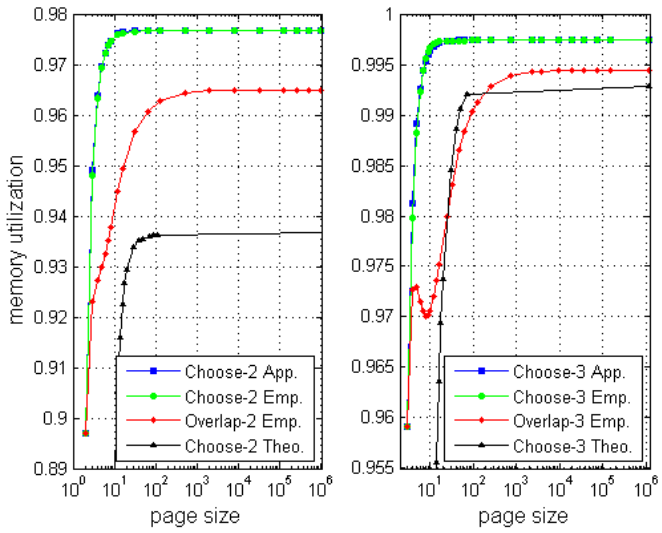

FIGURE 4.1.
Memory utilization vs. page size. Empirical CUCKOO-CHOOSE-K (green), Empirical CUCKOO-OVERLAP (red), approximation formula of Empirical CUCKOO-CHOOSE-K (blue), and theoretical lower bound of CUCKOO-CHOOSE-K (black). left: $k = 2$, right: $k = 3$.

$$N\left(\underline{a}\right) = \binom{g}{a_0, ..., a_t} \prod_{i=0}^{t} \binom{t}{i}^{a_i} \tag{3.5}$$

For the asymptotic behavior of $N\left(\underline{a}\right)$, we are going to use the following lemma:

Lemma 9. $\binom{g}{a_0,...,a_t} \leq \prod_{i=0}^{t} \left(\frac{g}{a_i}\right)^{a_i}$

Proof. Details in the full version. □

Theorem 10. *Let*
$c_9\left(\hat{a}, \beta\right) = \prod_{i=0}^{t} \left(\frac{1}{\hat{a}_i}\right)^{\frac{\hat{a}_i}{t}} \prod_{i=0}^{t} \binom{t}{i}^{\frac{\hat{a}_i}{t}} \left(\frac{\beta}{\beta - x}\right)^{(\beta - x)} \left(\frac{\beta}{x}\right)^{x} p_{hit}^{x} \left(1 - p_1 - p_{hit}\right)^{\beta - x}$. *If* $c_9\left(\hat{a}, \beta\right) < 1 - \epsilon$ *for all* $\forall x_1 \leq x < \beta$, *then* $P_{bad}\left(\hat{a}, \beta\right)$ *decreases exponentially as* $n \to \infty$. *Any memory utilization* β *that satisfies the constraint is a lower bound on the possible memory utilization.*

Proof. Details in the full version. □

Theoretical lower bounds of the memory utilization obtained from numerical solutions of the constraint $c_9\left(\hat{a}, \beta\right) < 1$ are displayed in figure 4.1.

4. Empirical Results

The experiments were conducted with a similar protocol to the one described in [2]. In all experiments the number of the buckets, d, was two. The capacity of each bucket k was either two or three. The size of the hash tables n was $1,209,600$. The reported memory utilization β is the mean memory utilization over twenty trials. The random

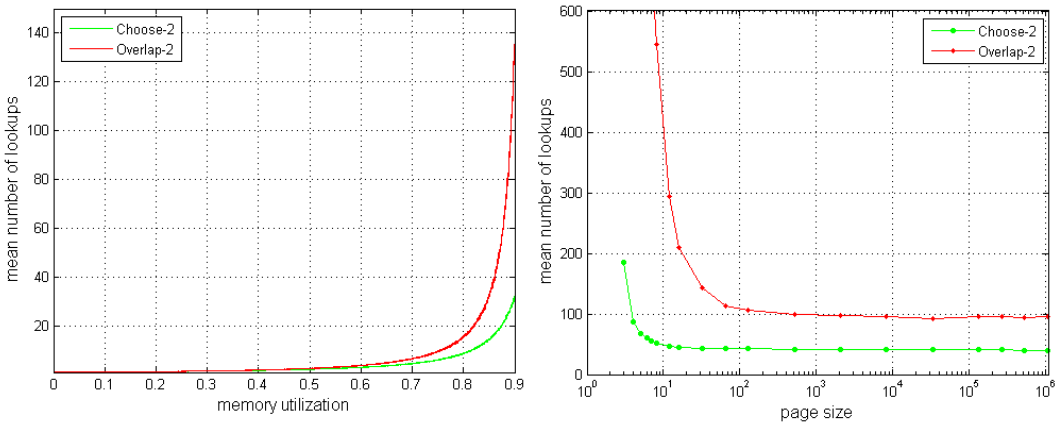

FIGURE 4.2.
Number of lookups required to insert an item vs. memory utilization (left) and vs. page size (right). In the left figure the page size is 8. In the right figure the memory utilization is 92%. The left figure was smoothed with an averaging filter. The variance in the number of lookups required to insert an item was much smaller in CUCKOO-CHOOSE-K.

hash functions were based on the Matlab "rand" function with the twister method. Items were inserted into the hash table one-by-one until an item could not be inserted. The results of both CUCKOO-CHOOSE-K and CUCKOO-OVERLAP were notably stable. In each case, the standard deviation was a few hundredths of a percent, so error bars would be invisible in the figure. Such strongly predictable behavior is appealing from a practical standpoint. Since we added a paging constraint, our results are not comparable to previous works that do not include a paging constraint.

Experiments show that CUCKOO-CHOOSE-K improves memory utilization significantly even for a small page size t and a small bucket capacity k when compared to the classical cuckoo hashing CUCKOO-DISJOINT. It outperforms CUCKOO-OVERLAP as well. Recall that CUCKOO-DISJOINT is the extreme case of CUCKOO-CHOOSE-K and CUCKOO-OVERLAP when the page size is equal to bucket size k. The memory utilization $\beta_{Choose-k}$ converges very quickly to its maximum value. For example $\beta_{Choose-2}(t = 16) = 0.9763$ is almost equal to $\beta_{Choose-2}(t = 1,209,600) = 0.9767$. Whereas $\beta_{Overlap-2}(t = 16) = 0.9494$ and $\beta_{Disjoint-2} = 0.8970$.

The empirical memory utilization can be approximated very accurately by the following formulas:

$$\beta_{Choose-2}(t) \approx 0.977 \cdot \left(1 - (\frac{t}{0.764})^{-2.604}\right) \tag{4.1}$$

$$\beta_{Choose-3}(t) \approx 0.997 \cdot \left(1 - (\frac{t}{1.011})^{-2.998}\right) \tag{4.2}$$

The maximum approximation error is 0.0011 for $k = 2$ and 0.0015 for $k = 3$.

The empirical results and their approximations are displayed in figure 4.1 together with empirical results of CUCKOO-OVERLAP. The memory utilization for $kd > 6$ rapidly approaches one (not displayed).

CUCKOO-CHOOSE-K outperforms CUCKOO-OVERLAP not only in memory utilization, but in the number of iterations required to insert a new item as well. Figure 4.2 illustrates the number of iterations required to insert a new item when the hash table is 92% full and we use breadth first search to search for a vacant position. $\#itr_{Choose-2}(t = 8)$ =52, whereas $\#itr_{Overlap-2}(t = 8) = 545$. Note that in these simulations we did not limit the number of insert iterations and we continued to insert items as long as we could find free locations. Most applications limit the number of inserted iterations, and maintain a low memory utilization in order to find a free location easily. If an empty position is not found within a fixed number of iterations, a rehash is performed or the item is placed outside of the cuckoo array.

References

[1] M. Dietzfelbinger, A. Goerdt, M. Mitzenmacher, A. Montanari, R. Pagh, and M. Rink, "Tight thresholds for cuckoo hashing via XORSAT". ICALP (1) 2010: 213-225

[2] E. Lehman and R. Panigrahy, "3.5-way cuckoo hashing for the price of 2 and a bit". In Proceedings of the 17th Annual European Symposium on Algorithms, pages 671–681, 2009.

[3] M. Dietzfelbinger and P. Woelfel, "Almost random graphs with simple hash functions", 35th STOC, pages 629–638, 2003.

[4] R. Pagh and F. Rodler. Cuckoo hashing. Journal of Algorithms 51 (2004), p. 122-144.

[5] P. Sanders, D. Fotakis, R. Pagh and P. Spirakis. "Space efficient hash tables with worst case constant access time". Theory of computing systems 38, 229-248 (2005).

[6] R. Panigrahy, "Efficient hashing with lookups in two memory accesses", SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, pages 830–839, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[7] D. Fernholz and V. Ramachandran. "The k-orientability thresholds for $G_{n,p}$". In SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pages 459–468, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[8] J. A. Cain, P. Sanders and N. Wormald, "The random graph threshold for k-orientiability and a fast algorithm for optimal multiple-choice allocation". In SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pages 469–476, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[9] A. Kirsch, M. Mitzenmacher, and U. Wieder. "More robust hashing: Cuckoo hashing with a stash". SIAM Journal on Computing 39:1543-1561, 2009.

[10] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. SIAM Journal on Computing, 29:180-200, 1999.A preliminary version of this paper appeared in Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing, 1994.

[11] Fountoulakis, N., Khosla, M., Panagioutou, K.: The Multiple-orientability Thresholds for Random Hypergraphs. In: Proc. 22nd SODA. SIAM (2011)

[12] A. Frieze, P. Melsted, Maximum Matchings in Random Bipartite Graphs and the Space Utilization of Cuckoo Hashtables, CoRR abs/0910.5535: (2009)

[13] Fountoulakis, Panagiotou, Orientability of Random Hypergraphs and the Power of Multiple Choices, ICALP (1) 2010: 348-359

[14] M. Dietzfelbinger, M. Mitzenmacher and M. Rink, "Cuckoo Hashing with Pages", arXiv:1104.5111v1 [cs.DS], 2011