Cuckoo Hashing with a Stash: Alternative Analysis, Simple Hash Functions

Martin Dietzfelbinger

Technische Universität Ilmenau

Joint work with: Martin Aumüller
Cuckoo Hashing

Maintain a dynamic dictionary for $n$ keys

- lookups: $O(1)$
- deletions: $O(1)$
- insertions: $O(1)$ amortized expected
- space: $2(1 + \epsilon)n$ slots

Not so good: Insertion of a key set of size $n$ fails and rebuilds the whole data structure with probability $O(n^{-1})$. 

© Per H. Olsen
Motivation (Kirsch, Mitzenmacher, Wieder [KMW09])

In some applications, e.g.,
- high-performance routing (packet statistics)
- database indexing

a failure probability of $O(n^{-3})$ could already lead to a failure rate that is too high.

$\Rightarrow$ Cuckoo hashing not applicable, although its performance is suitable for such applications.

Task

Preserve the performance and lower the failure probability.
Kirsch, Mitzenmacher and Wieder [KMW09]:
- add a small constant-sized piece of memory, the so-called stash
- move elements that cannot be inserted to this stash

They prove: Using a stash of size \( s \) lowers failure probability from

\[
\mathcal{O}(n^{-1}) \text{ to } \mathcal{O}(n^{-(s+1)}).
\]

Proof is technically involved ("Poissonization", "Markov Chain coupling"). Assumes fully random hash functions.
New

- a different (simpler?) proof for the reduced failure probability of cuckoo hashing with a stash
- proof that a weak $O(1)$-time hash class is strong enough for c.h. with a stash
- more experimental evaluation
The Cuckoo Graph

The cuckoo graph $G(S, h_1, h_2)$:

- an undirected bipartite multigraph $(L, R, E)$ where $L$ and $R$ represent the table cells
- $E = \{(h_1(x), h_2(x)) \mid x \in S\}$

Provides us with information about

- if a rehash will occur during the insertion of $S$
- how long insertions will take
The Cuckoo Graph - Example
Question: Will all key insertions be successful?

Lemma (Devroye, Morin [DM03])

The hash functions $h_1$ and $h_2$ successfully insert all keys in $S$ if and only if each connected component of $G(S, h_1, h_2)$ is either a tree or unicyclic.

Answer: No.
Resolving Failures by Using a Stash

When the insertion procedure loops, the reason is that a (minimal) subgraph with more edges than nodes has been found. Move (an arbitrary) one of the edges of this subgraph to the stash, resolving the problem.

Figure: The two possible minimal structures of subgraphs with more edges than nodes.

After the key is removed, a unicyclic component remains, the graph is suitable for cuckoo hashing again.
The Size of the Stash

How many items are stored in the stash after inserting \( n \) keys?

**Definition**

The excess \( \text{ex}(G) \) is the minimal number of edges we have to remove from \( G \) such that all connected components in \( G \) contain at most one cycle.

**Proposition (Kirsch et al. [KMW09])**

After the insertion of \( S \) there are exactly \( \text{ex}(G(S, h_1, h_2)) \) keys in the stash.
The Size of the Stash – Example
The Size of the Stash – Example
The Size of the Stash – Example
The Size of the Stash – Example

We assume stash of small constant size $s$.

Central Question
How likely is it that more than $s$ keys are moved into the stash?

Equivalent question: $\Pr(\text{ex}(G) > s) = ?$
Main Theorem

Theorem 1 (Kirsch et al. [KMW09], here: new proof)

Let $G = G(S, h_1, h_2)$ be the random cuckoo graph on key set $S \subseteq U$ and fully random hash functions $h_1, h_2$ from $U$ to $[m]$, where $|S| = n$ and $m = (1 + \varepsilon)n$ for an $\varepsilon \in (0, 1)$. Then

$$\Pr(\text{ex}(G) > s) = \mathcal{O}\left(n^{-(s+1)}\right).$$

Related work: Kutzelnigg [Kut09] obtained the constant in the $\mathcal{O}\left(n^{-(s+1)}\right)$ expression. Technically involved (generating functions, saddle point method), needs fully random hash functions.
Part 1: New Proof (based on [DW03])

We know: If stash size $s$ is not sufficient, then $\text{ex}(G(S, h_1, h_2)) > s$.

Idea: Concentrate on subgraph with excess $s + 1$.

Definition

An excess-$(s + 1)$ core structure of $G = G(S, h_1, h_2)$ is a subgraph $G'$ of $G$ with the following properties:

1. $G'$ has excess exactly $s + 1$.
2. $G'$ has no leaf edges.
3. $G'$ contains only components with at least two cycles.

Pretty obvious: Stash of size $s$ overflows $\iff$ cuckoo graph contains an excess-$(s + 1)$ core structure.
Analysis of Stash Size – Example

Question: Stash size 2 sufficient?
Answer: No, we can find an excess-3 core structure.
Alternative Approach to Analysis
(used in D., Woelfel [DW03])

- **count** non-isomorphic graphs that form an excess-$(s + 1)$ core structure
- **bound probability** that one of the excess core structures is realized
Counting Graphs

Definition

Let \( N(k, \ell, q) \) denote the number of non-isomorphic connected graphs with \( k - \ell \) inner edges, \( \ell \) leaf edges and cyclomatic number \( q \).

Lemma (cf. [DW03])

\[
N(k, \ell, q) < \left( \frac{k^2q}{2} \right) \cdot (k - q)^{2\ell + 4q - 4}.
\]

- cyclomatic number \( \gamma(G) \) of \( G \) = minimum number of edges to remove to make \( G \) acyclic.
- know: \( \gamma(G) = \text{ex}(G) + \text{cc}(G) \), where \( \text{cc}(G) \) is the number of cyclic components.
Counting Graphs

Definition

Let $N(k, \ell, c, s)$ denote the number of non-isomorphic graphs with $k - \ell$ inner edges, $\ell$ leaf edges, $c$ connected components and excess $s$.

Lemma 1

$$N(k, \ell, c, s) < (k + c - s)^{2\ell + 6s + 8c - 6}$$

Proof.

Three steps:

- $N(k, \ell, 1, 0)$ – simple, use $N(k, \ell, 0) + N(k, \ell, 1)$.
- $N(k, \ell, 1, s)$ – remove $s$ edges such that remaining graph has excess 0. $(k - s)^{2s}$ choices for endpoints, use $N(k, \ell, 1, 0)$.
- $N(k, \ell, c, s)$ – add $c - 1$ edges to connect the graph, $(k - s + c)^{2(c-1)}$ choices for endpoints, use $N(k, \ell, 1, s)$ (actually more tricky.)
Probability for Excess Core Structures

Let $K(T) = G(T, h_1, h_2)$, $T \subseteq S$, denote the subgraph of $G$ consisting of all edges for keys $x \in T$, disregarding isolated vertices.

**Lemma 2**

Let $T \subseteq U$, and $H = (V_H, E_H)$ be a bipartite graph, edges uniquely labeled with the elements of $T$. If values $h_i(x)$ are chosen fully randomly for all $x \in T, i \in \{1, 2\}$, then the probability that $K(T)$ is isomorphic to $H$ is

$$2^c \cdot m^{-|E_H|-\gamma(H)+c},$$

where $c$ denotes the number of connected components of $H$.

[DW03] proved this for connected graphs. Result can be obtained by multiplying over connected components of an arbitrary graph. (Easy.)
Proof of Theorem 1

If $\text{ex}(G) > s$, then

- there exists $T \subseteq S, |T| = k$, such that $K(T)$ forms an excess-$(s + 1)$ core structure.
- all components of $K(T)$ are cyclic, hence $\text{ex}(K(T)) = \gamma(K(T)) - c$
- isomorphism probability: $2^c \cdot m^{-k-s-1}$

Now: Short calculation.
Proof of Theorem 1

\[
\Pr(\text{ex}(G) > s) \leq \sum_{k=s+3}^{n} \sum_{c=1}^{s+1} \frac{2^c \cdot n^k \cdot N(k, 0, c, s + 1)}{m^{k+s+1}}
\]

\[
< \sum_{k=s+3}^{n} \sum_{c=1}^{s+1} \frac{2^c \cdot n^k \cdot (k - s - 1 + c)^{6(s+1)+8c-6}}{((1 + \varepsilon)n)^{k+s+1}}
\]

\[
= \frac{1}{n^{s+1}} \sum_{k=s+3}^{n} \sum_{c=1}^{s+1} \frac{2^c \cdot (k - s - 1 + c)^{6s+8c}}{(1 + \varepsilon)^{k+s+1}}
\]

\[
\leq \frac{(s + 1)2^{s+1}}{n^{s+1}} \sum_{k=s+3}^{n} \frac{k^{O(1)}}{(1 + \varepsilon)^{k+s+1}} = O(n^{-(s+1)}).
\]
Part 2: “Realistic” Hash Functions

Analysis from [KMW09] and [Kut09] requires fully random hash functions: not so easily come by in practice.

**Question**

Analysis adaptable using hash functions with a bounded degree of independence, e.g., $d$-wise independent hash functions, which can be efficiently evaluated (like polynomials of degree $d - 1$)?

**Alternative:**

- Mitzenmacher, Vadhan ([MV08]), Chung, Vadhan([CV08]): some entropy in keys + 1-universal class $\rightarrow$ close to random behavior
- some risks w.r.t. cuckoo hashing (D., Schellbach [DS09])
Class of Hash Functions [DW03]

- $g : U \rightarrow [r]$ from $d$-wise independent class
- $f_1, f_2 : U \rightarrow [m]$ from $d$-wise independent class
- $z_0^{(1)}, \ldots, z_{r-1}^{(1)}$ and $z_0^{(2)}, \ldots, z_{r-1}^{(2)}$ random from $[m]$, tabulated

Hash functions:

\[
h_1(x) = \left( f_1(x) + z_{g(x)}^{(1)} \right) \mod m
\]
\[
h_2(x) = \left( f_2(x) + z_{g(x)}^{(2)} \right) \mod m
\]

Evaluation in constant time! Class of these hash functions: $\hat{R}_{r,m}^d$. 
Theorem 2

Let $T \subseteq U$. Let $|g(T)| \geq |T| - \ell$ for $(h_1, h_2) \in \hat{R}^{2\ell}_{r,m}$. Then all $(h_1(x), h_2(x)), x \in T$, are uniformly and independently distributed in $[m]^2$. 
Theorem 2

Let $T \subseteq U$. Let $|g(T)| \geq |T| - \ell$ for $(h_1, h_2) \in \hat{R}_{r,m}^{2\ell}$. Then all $(h_1(x), h_2(x)), x \in T$, are uniformly and independently distributed in $[m]^2$.

- only keys in $T'_1$ and $T_2$ collide under $g$, at most $2\ell$ colliding keys
- $f$ compensates for these collisions
- displacements: fully random
Full Randomness on Excess Core Structures

We need full randomness on excess-\((s + 1)\) core structures to reuse previous analysis.

- **Define** \(G(S, h_1, h_2)\) to be \(\ell\)-bad if there exists \(T \subseteq S\) with \(|g(T)| < |T| - \ell\) and \(K(T) = G(T, h_1, h_2)\) forms an excess core structure for excess \(s + 1\).
- **Then:** If \(G(S, h_1, h_2)\) is “good” then hash function pair works fully randomly on all excess core structures of our interest \(\Rightarrow\) can reuse analysis!
- **Question:** How likely is it that \(G(S, h_1, h_2)\) is good?
Bounding Probability of $\ell$-bad Graphs

**Problem:** Pair of hash functions does not work fully randomly on bad graphs, because $|g(T)| < |T| - \ell$.

- works fully randomly on all $T' \subset T : |g(T')| \geq |T'| - \ell$
- extract subgraph $K(T')$ with $|g(T')| = |T'| - \ell$, so-called $2\ell$-reduced subgraph
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 2$. 

Martin Dietzfelbinger (TU Ilmenau)  
Cuckoo Hashing with a Stash
Approach: \( G \) is \( \ell \)-bad \( \Rightarrow \) graphs contains an excess core structure \( K(T) \) with excess \( s + 1 \) and \( |g(T)| < |T| - \ell \).

Goal: \( |g(T)| = |T| - \ell \).

Status: \( |g(T)| = |T| - \ell - 2 \).

1. Mark all keys in \( K(T) \) that collide under \( g \).
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.
Goal: $|g(T)| = |T| - \ell$.
Status: $|g(T)| = |T| - \ell - 2$.

1. Mark all keys in $K(T)$ that collide under $g$.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 2$.

2. Remove unmarked components.
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 2$.

2. Remove unmarked components.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 2$.

3. **Remove components while** $|g(T)| \leq |T| - \ell$. 
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.
Goal: $|g(T)| = |T| - \ell$.
Status: $|g(T)| = |T| - \ell - 2$.

3. Remove components while $|g(T)| \leq |T| - \ell$. 

\[
\begin{align*}
\text{Diagram}
\end{align*}
\]
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 1$.

3. Remove components while $|g(T)| \leq |T| - \ell$. 
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 1$.

4. Cannot remove any further components. Concentrate on one component from now on.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 1$.

4. Cannot remove any further components. Concentrate on one component from now on.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 1$.

5. Remove one marked edge.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 1$.

5. Remove one marked edge.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell$.

5. Remove one marked edge.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell$.

6. Remove edges until all leaf and cycle edges are marked.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell$.

6. Remove edges until all leaf and cycle edges are marked.
Lemma 3

If $G$ is $\ell$-bad, then there exists a subset $T \subseteq S$ such that $|g(T)| = |T| - \ell$ and $K(T)$ has the following properties:

1. There is one connected component in $K(T)$ that has at most $2\ell$ leaf and cycle edges.
2. All other connected components do not have leaves.
3. There are at most $2\ell$ connected components.

To bound probability for $\ell$-bad subgraphs: Can now re-use counting approach and have an extra factor $\mathcal{O}(r^{-\ell})$ for the probability for the $g$-collisions to happen.
Cuckoo Hashing with a Stash and HF’s from class $\hat{R}$

**Theorem 3**

$$\Pr(G(S, h_1, h_2) \text{ is } \ell\text{-bad}) = \mathcal{O}(n \cdot r^{-\ell}).$$

For $r = n^\beta$, $\frac{1}{2} < \beta < 1$ and $\ell = 2(s + 2)$.

**Corollary**

$$\Pr(G(S, h_1, h_2) \text{ is } \ell\text{-bad}) = \mathcal{O}(n^{-(s+1)})$$
Conclusion

- stash of size $s$ reduces failure probability drastically
  \( (\mathcal{O}(n^{-1}) \rightarrow \mathcal{O}(n^{-(s+1)})) \): New proof.
- analysis valid for constant-time, \( o(n) \)-space class \( \hat{\mathcal{R}} \).
- experimental results, extending those of Kirsch et al., strengthening the message “stashes do help”.
- a stash size of only 9 helps us to almost completely avoid rehashes in practical scenarios.

**Open Question:** Generalized Cuckoo Hashing (> 2 hash functions) with weak hash classes?
Practical Stash Sizes

Success Rate of Cuckoo Hashing for Fixed Table Load of 49% and Different Table Sizes
Success Rate of Cuckoo Hashing for Table Size of 500 and Different Table Loads

Table Load in Percent

Success Rate in Percent

Stash 0
Stash 3
Stash 9
Success Rate of Cuckoo Hashing for Table Size of 50000 and Different Table Loads

Table Load in Percent: 40, 42.5, 45, 47, 48, 49, 49.5
Success Rate in Percent: 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100

Stash 0, Stash 3, Stash 9

Martin Dietzfelbinger (TU Ilmenau)
Thank you!
Kai-Min Chung and Salil P. Vadhan.  
Tight bounds for hashing block sources.  

Luc Devroye and Pat Morin.  
Cuckoo hashing: Further analysis.  

Martin Dietzfelbinger and Ulf Schellbach.  
On risks of using cuckoo hashing with simple universal hash classes.  
References II

Martin Dietzfelbinger and Philipp Woelfel.
Almost random graphs with simple hash functions.

Adam Kirsch, Michael Mitzenmacher, and Udi Wieder.
More robust hashing: Cuckoo hashing with a stash.

Reinhard Kutzelnigg.
A further analysis of cuckoo hashing with a stash and random graphs of excess $r$.
Manuscript, 2009.