Karlsruhe Institute of Technology

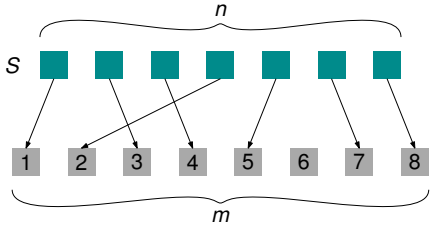
SicHash – Small Irregular Cuckoo Tables for Perfect Hashing

ALENEX 2023, Florence

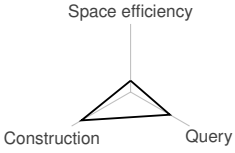
Hans-Peter Lehmann, Peter Sanders, Stefan Walzer | January 23, 2023

Perfect Hashing

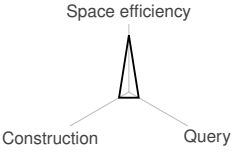
- Static set S of n objects
- Injectively map objects to the first m integers
- Load factor: $\alpha = m/n$
- Minimal Perfect Hashing: $\alpha = 1$
- Applications: databases, hash tables, approximate membership, retrieval, representatives
- Here: mainly non-minimal, load factors between 80% and 97%, but extension to minimal



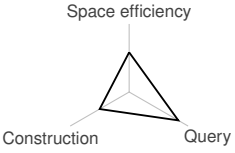
Motivation



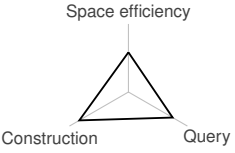
MeraculousHash [CHS+11]
FiPHa [MSSZ14]
BBHash [LRCP17]



RecSplit [EGV20]



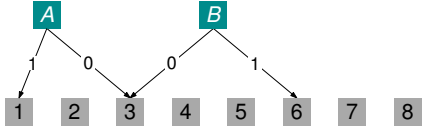
PTHash [PT21]



SicHash

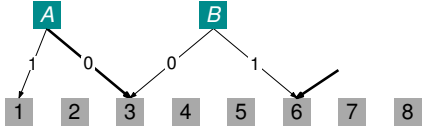
Cuckoo Hash Tables [PR04]

- Each object has two choices for cells
- On collision, move existing object
- d-ary: more than two choices [FPSS05]
  - Higher load factors
  - Insertion more complex
- Irregular: different objects have different number of hash functions [DGM+10]
  - Average number of hash functions $\hat{d}$
  - Best load factors by interpolating between $\lfloor \hat{d} \rfloor$ and $\lceil \hat{d} \rceil$



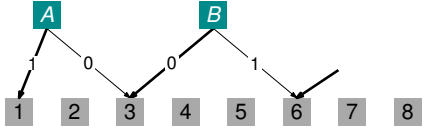
Cuckoo Hash Tables [PR04]

- Each object has two choices for cells
- On collision, move existing object
- d-ary: more than two choices [FPSS05]
  - Higher load factors
  - Insertion more complex
- Irregular: different objects have different number of hash functions [DGM+10]
  - Average number of hash functions $\hat{d}$
  - Best load factors by interpolating between $\lfloor \hat{d} \rfloor$ and $\lceil \hat{d} \rceil$



Cuckoo Hash Tables [PR04]

- Each object has two choices for cells
- On collision, move existing object
- d-ary: more than two choices [FPSS05]
  - Higher load factors
  - Insertion more complex
- Irregular: different objects have different number of hash functions [DGM+10]
  - Average number of hash functions $\hat{d}$
  - Best load factors by interpolating between $\lfloor \hat{d} \rfloor$ and $\lceil \hat{d} \rceil$

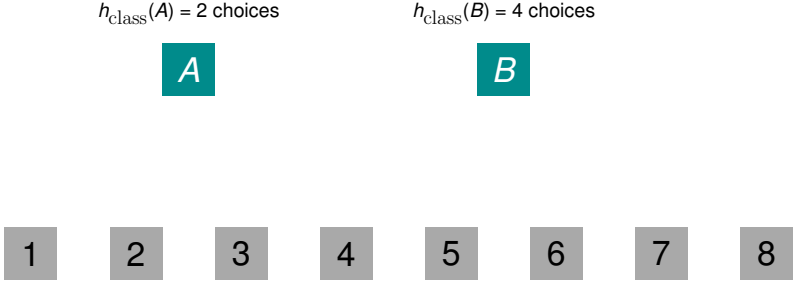


Perfect Hashing by Retrieval [DHSW22]

- Each object has 2^k choices
- Find collision-free mapping
- Store static function $S \to \{0,1\}^k$ in retrieval data structure
- Space: $\mathrm{O}(kn)$ bits
- No known implementation
  - Low load factors (see later slide)
  - Retrieval only recently efficient enough ($1.02 \cdot kn$ bits) [DHSW22]

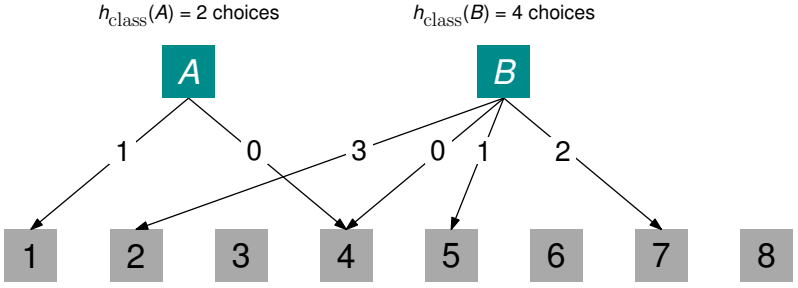
SicHash

- Combine Perfect Hashing by Retrieval [DHSW22] and Irregular Cuckoo Hashing [DGM+10]



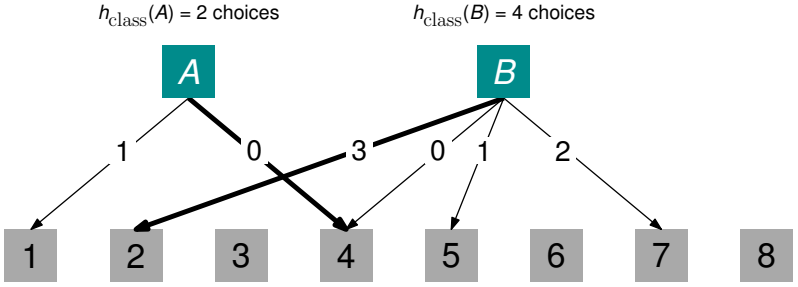
SicHash

- Combine Perfect Hashing by Retrieval [DHSW22] and Irregular Cuckoo Hashing [DGM+10]



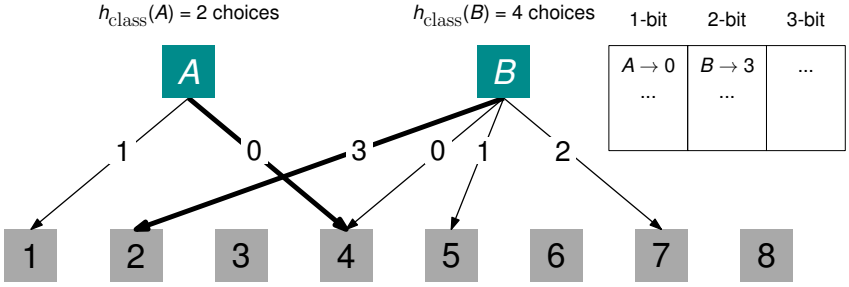
SicHash

- Combine Perfect Hashing by Retrieval [DHSW22] and Irregular Cuckoo Hashing [DGM+10]

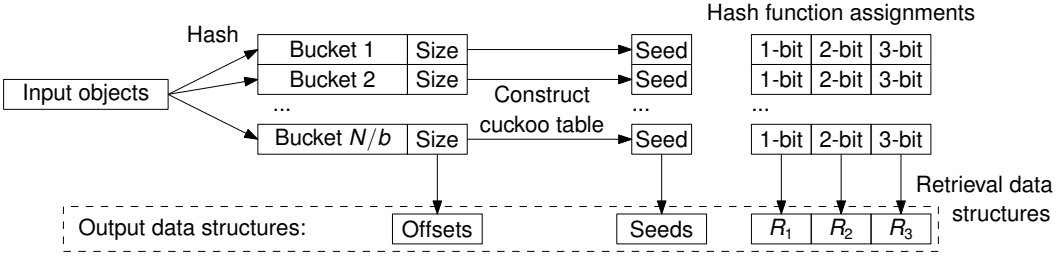


SicHash

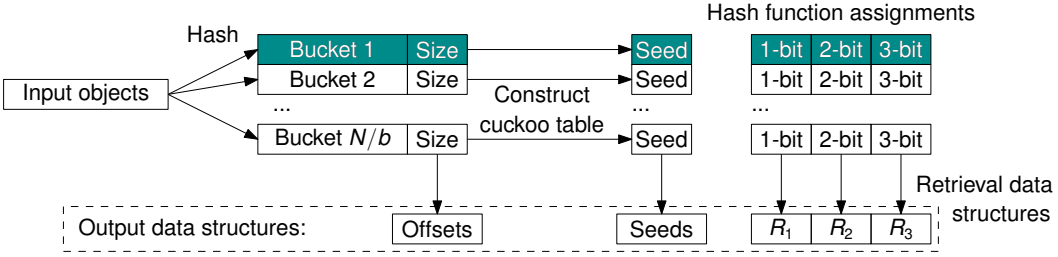
- Combine Perfect Hashing by Retrieval [DHSW22] and Irregular Cuckoo Hashing [DGM+10]



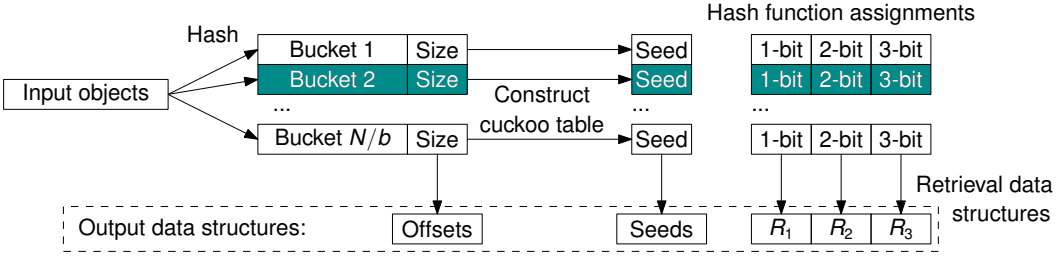
SicHash


Hash function assignments
Hash
Input objects
Bucket 1
Size
Bucket 2
Size
...
Bucket N/b
Size
Construct
cuckoo table
Seed
Seed
...
Seed
1-bit
2-bit
3-bit
1-bit
2-bit
3-bit
...
1-bit
2-bit
3-bit
Retrieval data
structures
Output data structures:
Offsets
Seeds
R_1
R_2
R_3

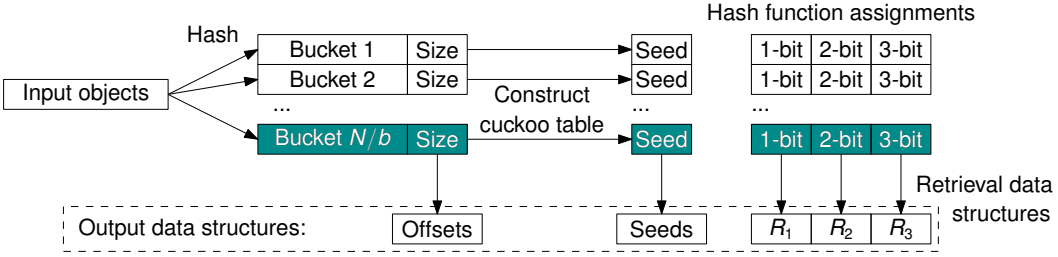
SicHash


Hash function assignments
Hash
Input objects
Bucket 1
Size
Bucket 2
Size
...
Bucket N/b
Size
Construct
cuckoo table
Seed
Seed
...
Seed
1-bit
2-bit
3-bit
1-bit
2-bit
3-bit
...
1-bit
2-bit
3-bit
Retrieval data
structures
Output data structures:
Offsets
Seeds
R_1
R_2
R_3

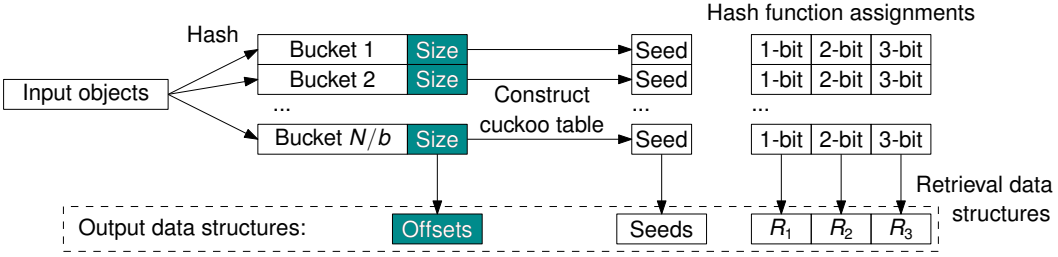
SicHash


Hash function assignments
Hash
Input objects
Bucket 1
Size
Bucket 2
Size
...
Bucket N/b
Size
Construct
cuckoo table
Seed
Seed
...
Seed
1-bit
2-bit
3-bit
1-bit
2-bit
3-bit
...
1-bit
2-bit
3-bit
Retrieval data
structures
Output data structures:
Offsets
Seeds
R_1
R_2
R_3

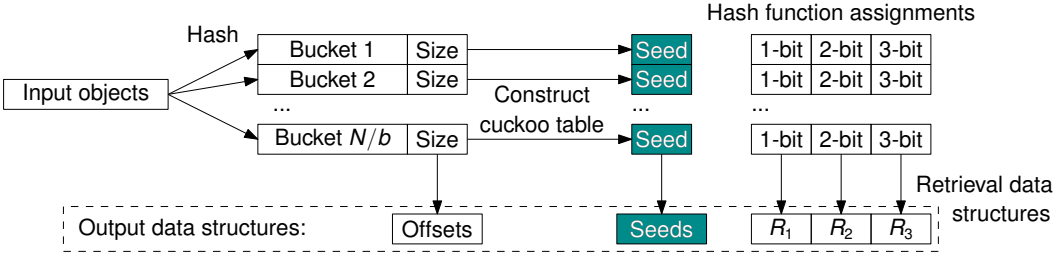
SicHash


Hash function assignments
Hash
Input objects
Bucket 1
Size
Bucket 2
Size
...
Bucket N/b
Size
Construct
cuckoo table
Seed
Seed
...
Seed
1-bit
2-bit
3-bit
1-bit
2-bit
3-bit
...
1-bit
2-bit
3-bit
Retrieval data
structures
Output data structures:
Offsets
Seeds
R_1
R_2
R_3

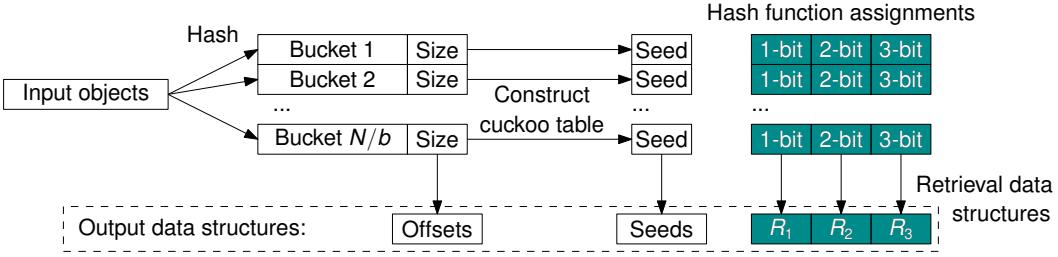
SicHash


Hash function assignments
Hash
Input objects
Bucket 1
Bucket 2
...
Bucket N/b
Size
Size
Size
Construct
cuckoo table
Seed
Seed
...
Seed
1-bit
2-bit
3-bit
1-bit
2-bit
3-bit
...
1-bit
2-bit
3-bit
Retrieval data
structures
Output data structures:
Offsets
Seeds
R_1
R_2
R_3

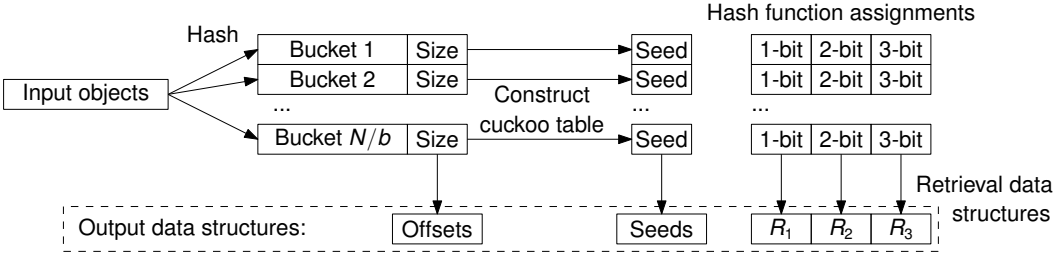
SicHash


Hash function assignments
Hash
Input objects
Bucket 1
Size
Bucket 2
Size
...
Bucket N/b
Size
Construct
cuckoo table
Seed
Seed
...
Seed
1-bit
2-bit
3-bit
1-bit
2-bit
3-bit
...
1-bit
2-bit
3-bit
Retrieval data
structures
Output data structures:
Offsets
Seeds
R_1
R_2
R_3

SicHash

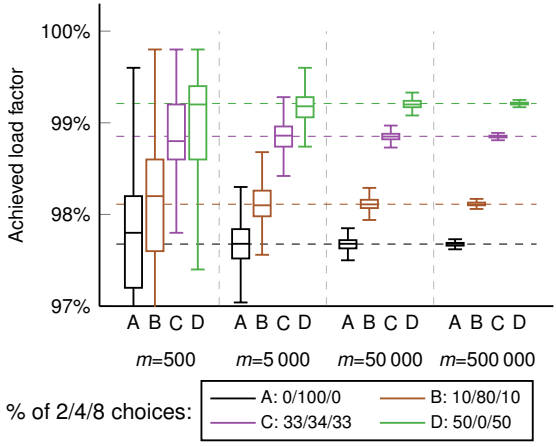

Hash function assignments
Hash
Input objects
Bucket 1
Size
Bucket 2
Size
...
Bucket N/b
Size
Construct
cuckoo table
Seed
Seed
...
Seed
1-bit
2-bit
3-bit
1-bit
2-bit
3-bit
...
1-bit
2-bit
3-bit
Retrieval data
structures
Output data structures:
Offsets
Seeds
R_1
R_2
R_3

SicHash

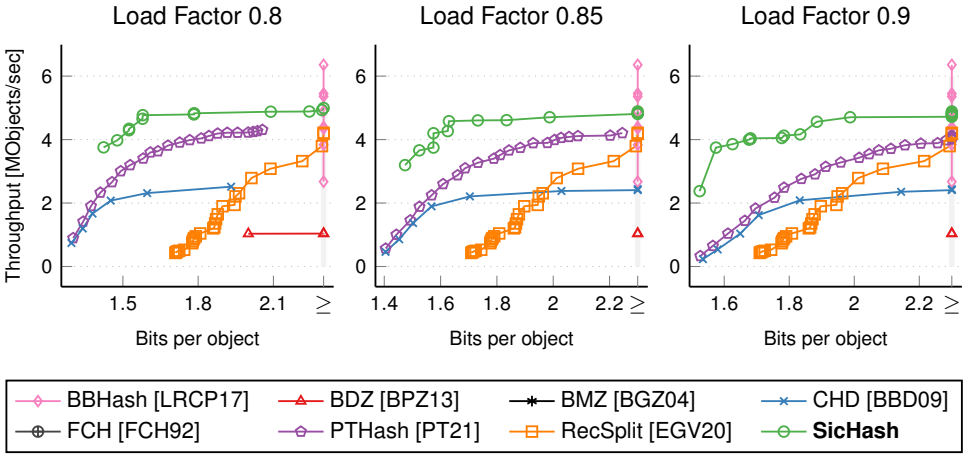

Hash function assignments
Hash
Input objects
Bucket 1
Size
Bucket 2
Size
...
Bucket N/b
Size
Construct
cuckoo table
Seed
Seed
...
Seed
1-bit
2-bit
3-bit
1-bit
2-bit
3-bit
...
1-bit
2-bit
3-bit
Retrieval data
structures
Output data structures:
Offsets
Seeds
R_1
R_2
R_3

SicHash

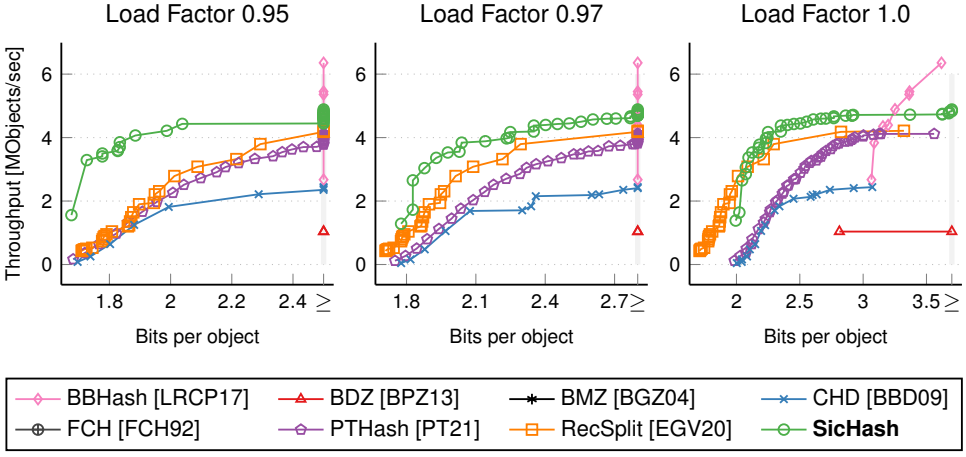
- Irregular cuckoo hashing [DGM$^+$10]: looking at average number of hash functions
  - Consecutive integers
- Here: looking at space usage
  - Average of $\log_2$(number hash functions)
  - Best load factors given by unbalanced distributions, not balanced ones!
- Overloading



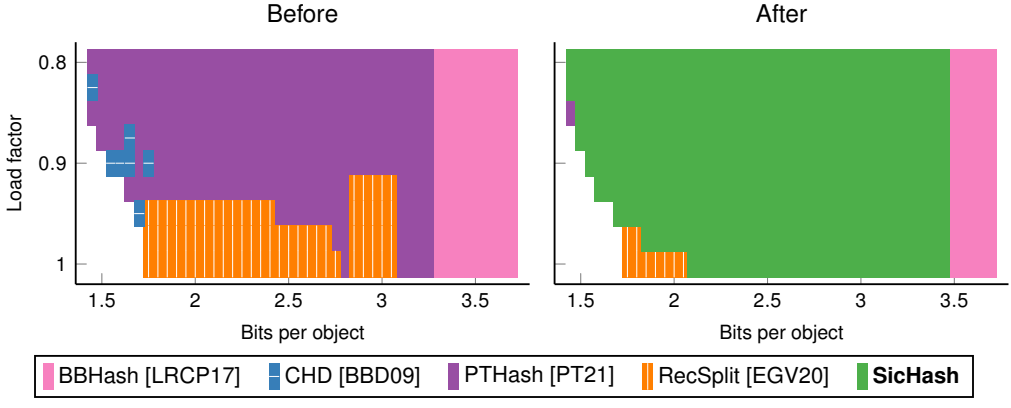
Construction Throughput



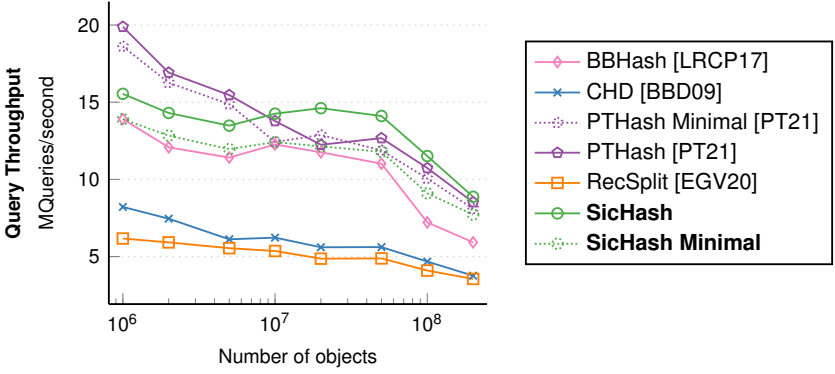
Construction Throughput



Best Construction Throughput



Query Throughput

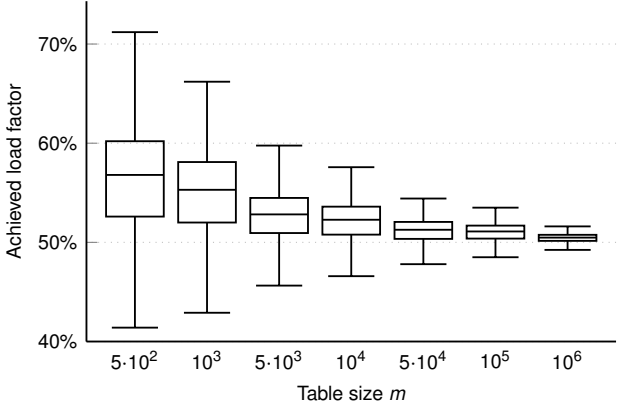


Conclusion

- SicHash Perfect Hash Function
- Combines retrieval and irregular cuckoo hashing
- Fastest construction for wide range of configurations
- Fast queries
- Code available under GPLv3 license:
  `https://github.com/ByteHamster/SicHash`
- Future work:
  - Parallelize
  - Analyze overloaded cuckoo hash tables
- New: SIMDRecSplit [BKLS22] with even faster construction but slow queries

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 882500).

Overloading binary cuckoo hash tables



References I


Djamal Belazzougui, Fabiano C. Botelho, and Martin Dietzfelbinger.
Hash, displace, and compress.
In *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 682–693. Springer, 2009.


Fabiano C Botelho, David M Gomes, and Nivio Ziviani.
A new algorithm for constructing minimal perfect hash functions.
differences, 100(2):09, 2004.


Dominik Bez, Florian Kurpicz, Hans-Peter Lehmann, and Peter Sanders.
High performance construction of recsplit based minimal perfect hash functions.
CoRR, abs/2212.09562, 2022.


Fabiano C. Botelho, Rasmus Pagh, and Nivio Ziviani.
Practical perfect hashing in nearly optimal space.
Inf. Syst., 38(1):108–131, 2013.

References II


Jarrod A. Chapman, Isaac Ho, Sirisha Sunkara, Shujun Luo, Gary P. Schroth, and Daniel S. Rokhsar.
Meraculous: De novo genome assembly with short paired-end reads.
PLOS ONE, 6(8):1–13, 08 2011.


Martin Dietzfelbinger, Andreas Goerdt, Michael Mitzenmacher, Andrea Montanari, Rasmus Pagh, and Michael Rink.
Tight thresholds for cuckoo hashing via XORSAT.
In *ICALP (1)*, volume 6198 of *Lecture Notes in Computer Science*, pages 213–225. Springer, 2010.


Peter C. Dillinger, Lorenz Hübschle-Schneider, Peter Sanders, and Stefan Walzer.
Fast succinct retrieval and approximate membership using ribbon.
In *SEA*, volume 233 of *LIPIcs*, pages 4:1–4:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

References III

- Emmanuel Esposito, Thomas Mueller Graf, and Sebastiano Vigna.
  Recsplit: Minimal perfect hashing via recursive splitting.
  In *ALENEX*, pages 175–185. SIAM, 2020.
- Edward A. Fox, Qi Fan Chen, and Lenwood S. Heath.
  A faster algorithm for constructing minimal perfect hash functions.
  In *SIGIR*, pages 266–273. ACM, 1992.
- Dimitris Fotakis, Rasmus Pagh, Peter Sanders, and Paul G. Spirakis.
  Space efficient hash tables with worst case constant access time.
  Theory Comput. Syst., 38(2):229–248, 2005.

References IV


Antoine Limasset, Guillaume Rizk, Rayan Chikhi, and Pierre Peterlongo.
Fast and scalable minimal perfect hashing for massive key sets.
In *SEA*, volume 75 of *LIPIcs*, pages 25:1–25:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.


Hans-Peter Lehmann, Peter Sanders, and Stefan Walzer.
Sichash - small irregular cuckoo tables for perfect hashing.
CoRR, abs/2210.01560, 2022.


Ingo Müller, Peter Sanders, Robert Schulze, and Wei Zhou.
Retrieval and perfect hashing using fingerprinting.
In *SEA*, volume 8504 of *Lecture Notes in Computer Science*, pages 138–149. Springer, 2014.

References V


Rasmus Pagh and Flemming Friche Rodler.
Cuckoo hashing.
J. Algorithms, 51(2):122–144, 2004.


Giulio E. Pibiri and Roberto Trani.
PTHash: Revisiting FCH minimal perfect hashing.
In *SIGIR*, pages 1339–1348. ACM, 2021.