MinCounter: An Efficient Cuckoo Hashing Scheme for Cloud Storage Systems

Yuanyuan Sun, Yu Hua, Dan Feng, Ling Yang, Pengfei Zuo, Shunde Cao

Wuhan National Laboratory for Optoelectronics

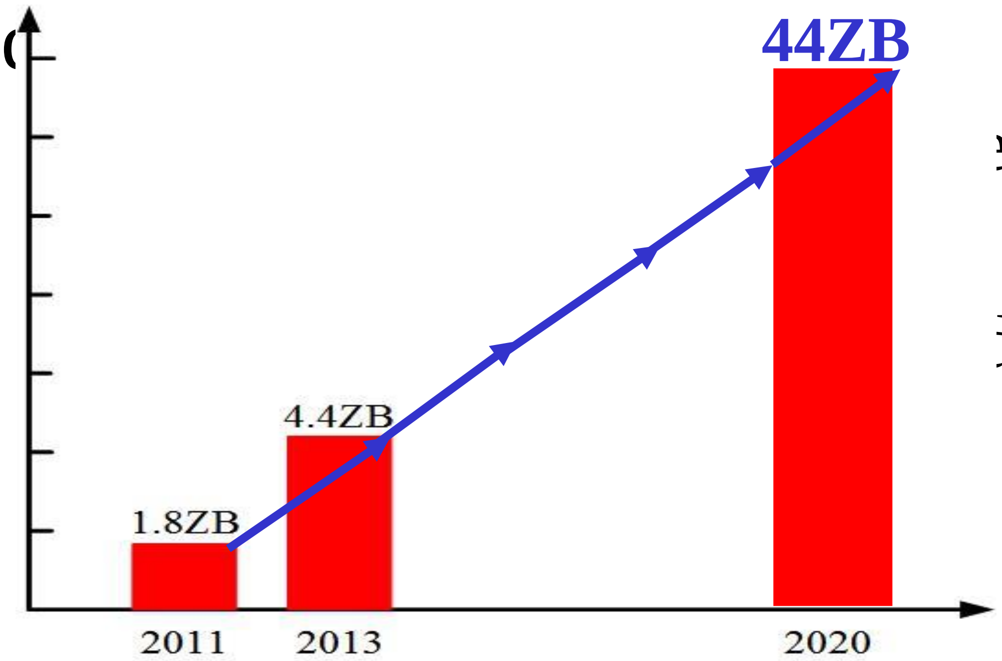
Huazhong University of Science and Technology

Outline

- Motivations and Backgrounds
- Design and Implementations
- Performance Evaluation
- Conclusion

Index for Big Data

- Large amounts of data (IDC)
  - 1.8ZB in 2011, 4.4ZB in 2013
  - 44ZB , 5.2TB for each user in 2020
- Inc
  - es
  -
  - l storage

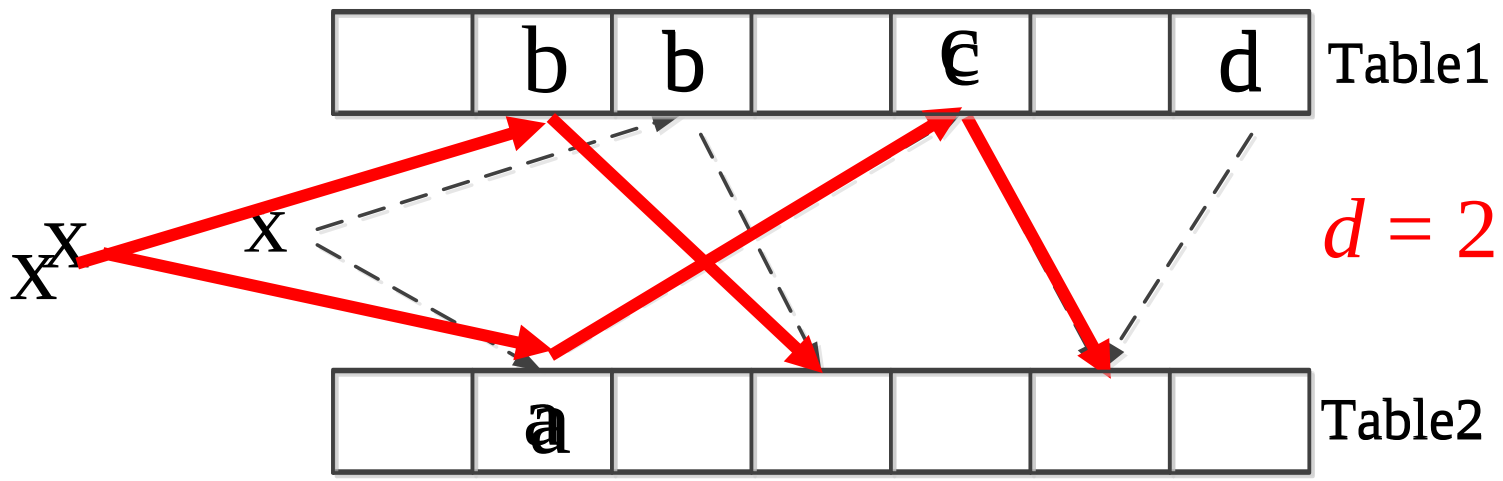


Hashing-based Index Structures

- Hashing-based data structures have been widely used in constructing the index.
- Advantages
  - Constant-scale addressing complexity
  - Fast query response
- Weaknesses
  - Low space utilization
  - High-latency risk of handling hashing collisions
- Cuckoo hashing

Cuckoo Hashing Scheme

- Uses *d* hash tables and *d* hash functions
- Random selection
- "Kicking-out" operation

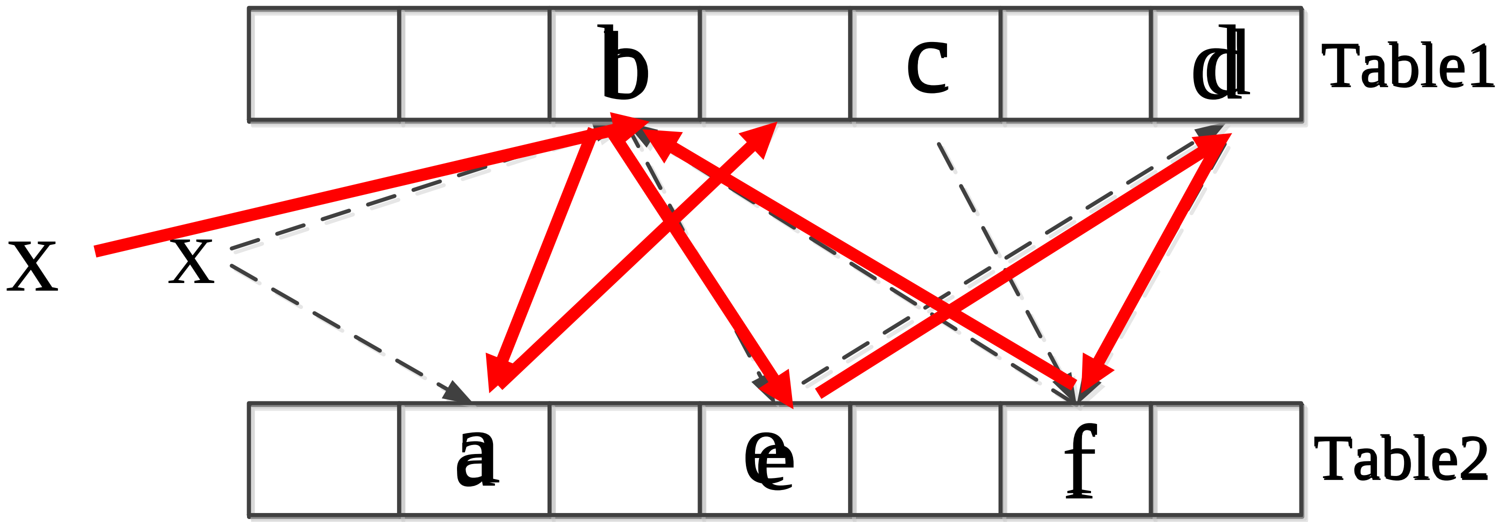


Advantages in Cuckoo Hashing

- Handle hash collisions
  - Moving the items among hash tables
- Ensure a more even distribution
  - *d* hash tables and *d* hash functions
- Constant-scale query time complexity
  - O(1)
- Improve space utilization

Challenges in Cuckoo Hashing

- Intensive migration operations
- Endless loops
  - Reconstruct hash tables

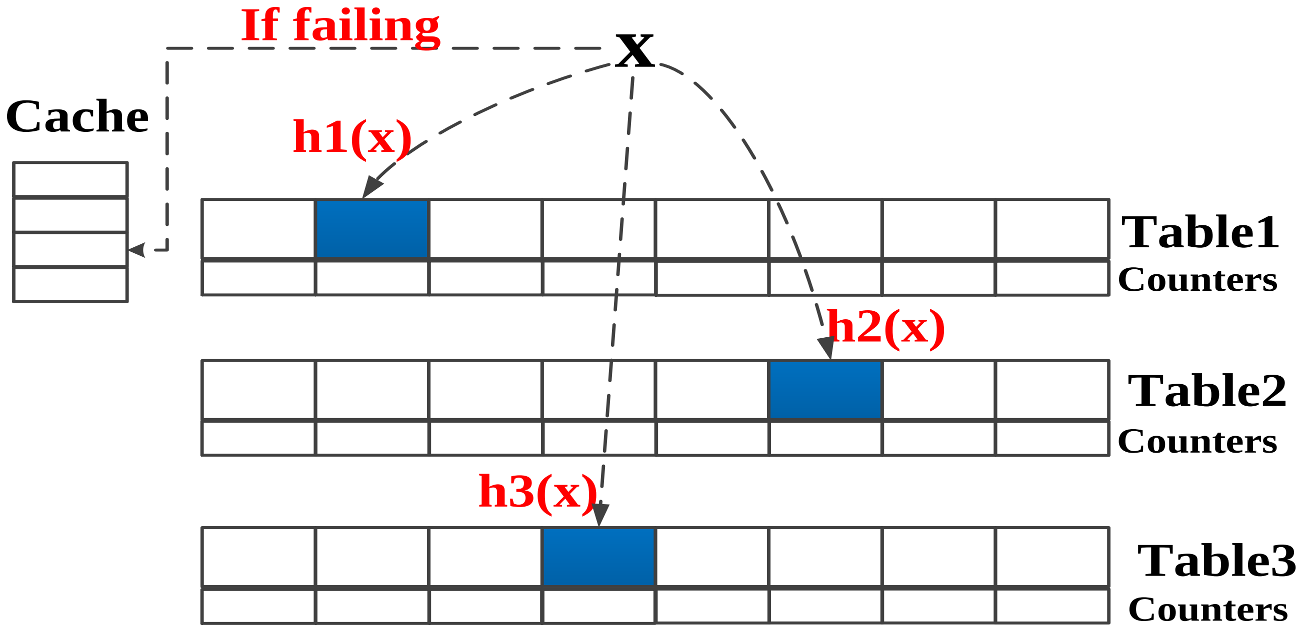


MinCounter

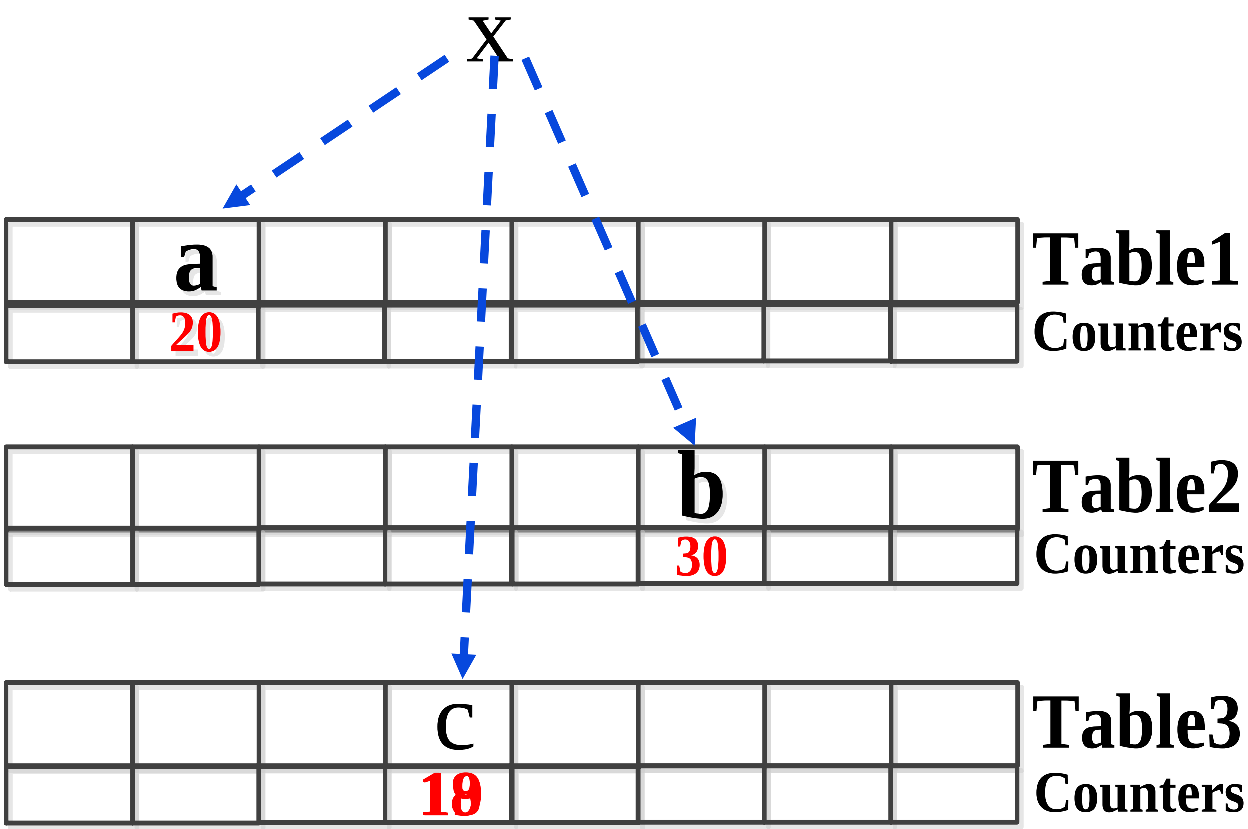
- Allocating a counter for each bucket to record kicking-out times
- Selecting the bucket with the minimum counter to kick out
- Avoiding busy routes and selecting the "cold" buckets
  - Infrequently accessed
  - Alleviate the occurrence of endless loops in data insertion process

Data Structure

- Focus on the cases of $d \geq 3$
- Insertion failure: kicking-out times is more than a threshold



An Example



x
a
20
Table1
Counters
b
30
Table2
Counters
c
10
Table3
Counters

Evaluation dataset

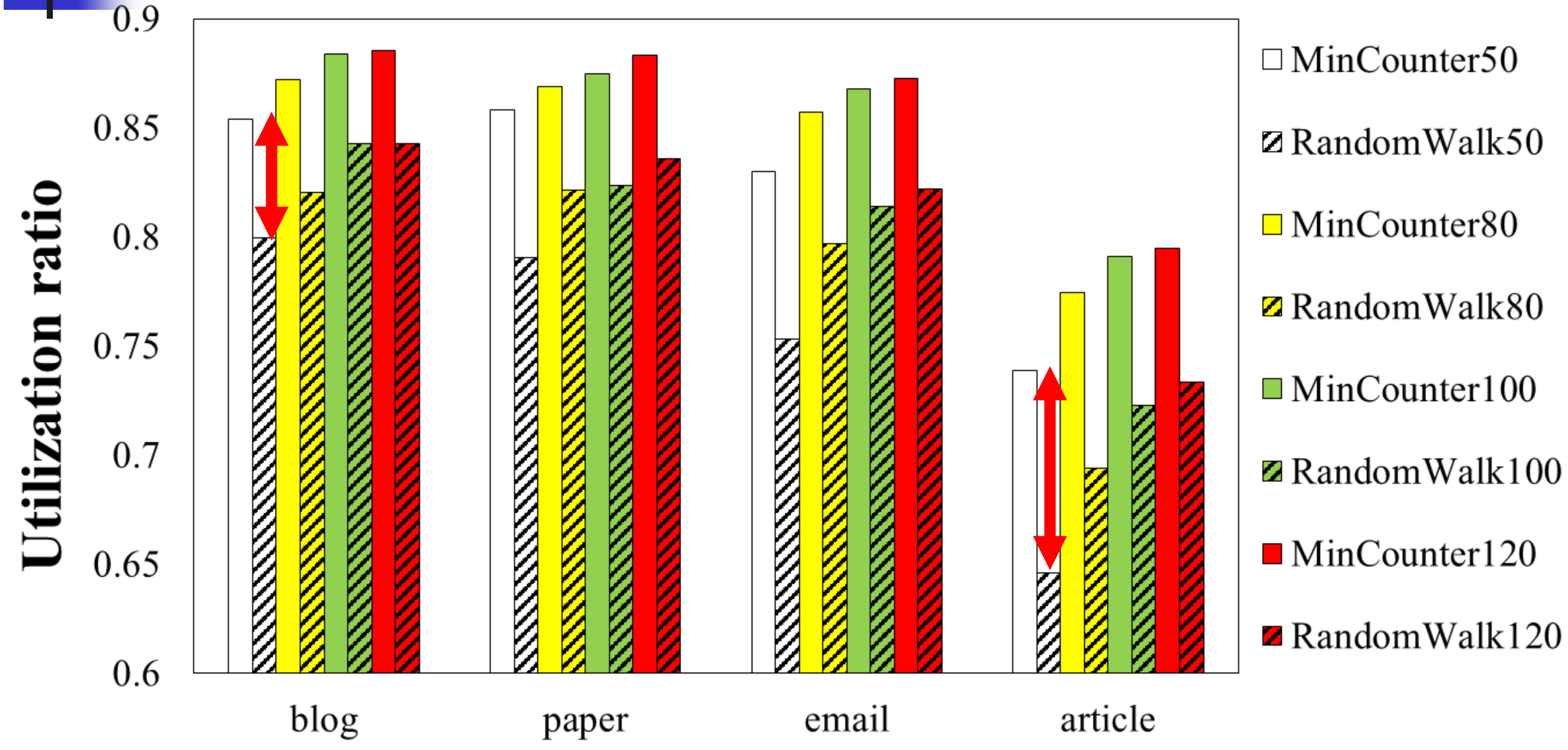
- Bag of Words
  - Four text collections in the form of bags-of-words
  - About 10 million items in total
  - Taking advantage of the union of docID and wordID as keys of items
  - http://archive.ics.uci.edu/ml/datasets/Bag+of+Words

Text collectons	Doc	Word	Total _W(Million)
KOS blog entries	3,430	6,906	0.4
NIFS full papers	1,500	12,419	0.8
Enron emails	39,861	28,102	3.8
NYTimes news articles	300,000	102,660	5.0

Evaluation Metrics

- Utilization ratio of hash tables
  - Occupied buckets / all buckets
  - Space efficiency
- Total kicking-out times during insertion operations
  - Insertion latency
- The kicking-out thresholds: 50, 80, 100 and 120
- The initial rate of hash tables: size of hash table / size of dataset
  - Initial rate 1.1: high collision rate
  - Initial rate 2.04: low collision rate

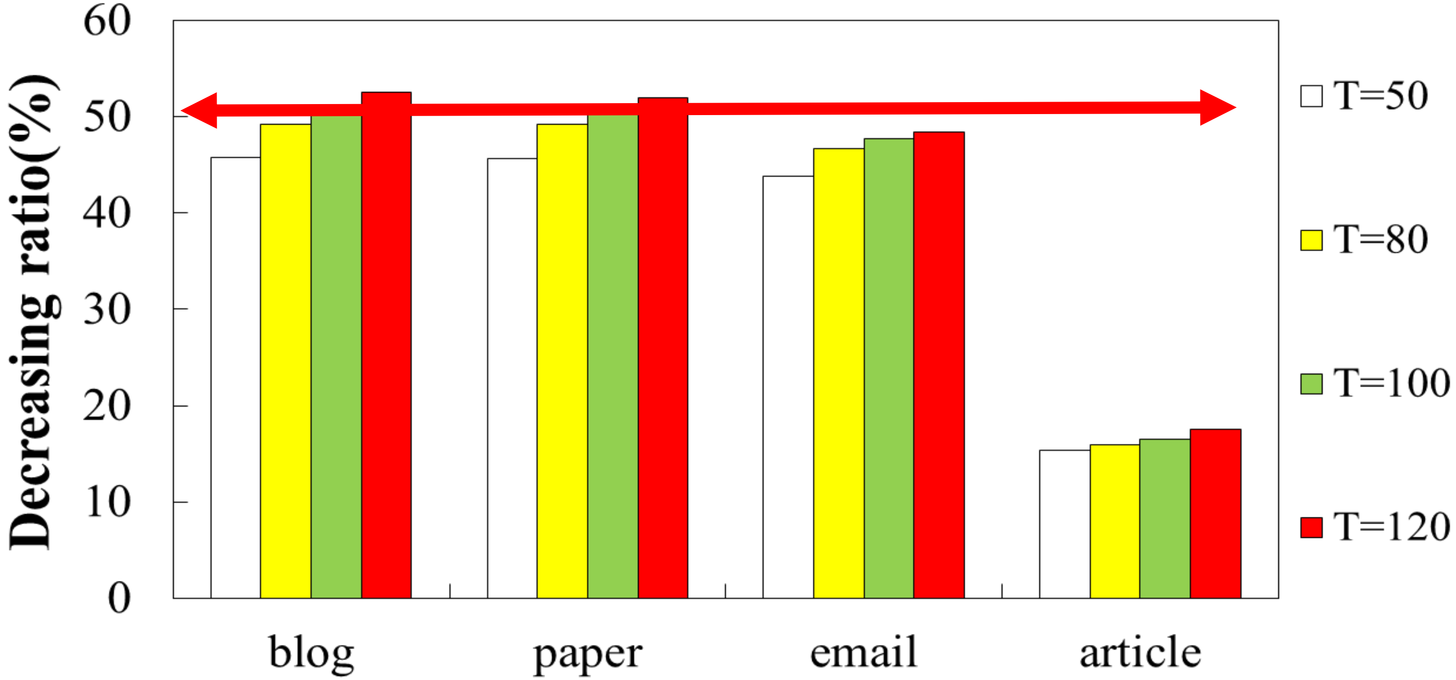
Utilization ratio of hash tables



- MinCounter obtains 5%-10% utilization improvement, compared with RandomWalk scheme.

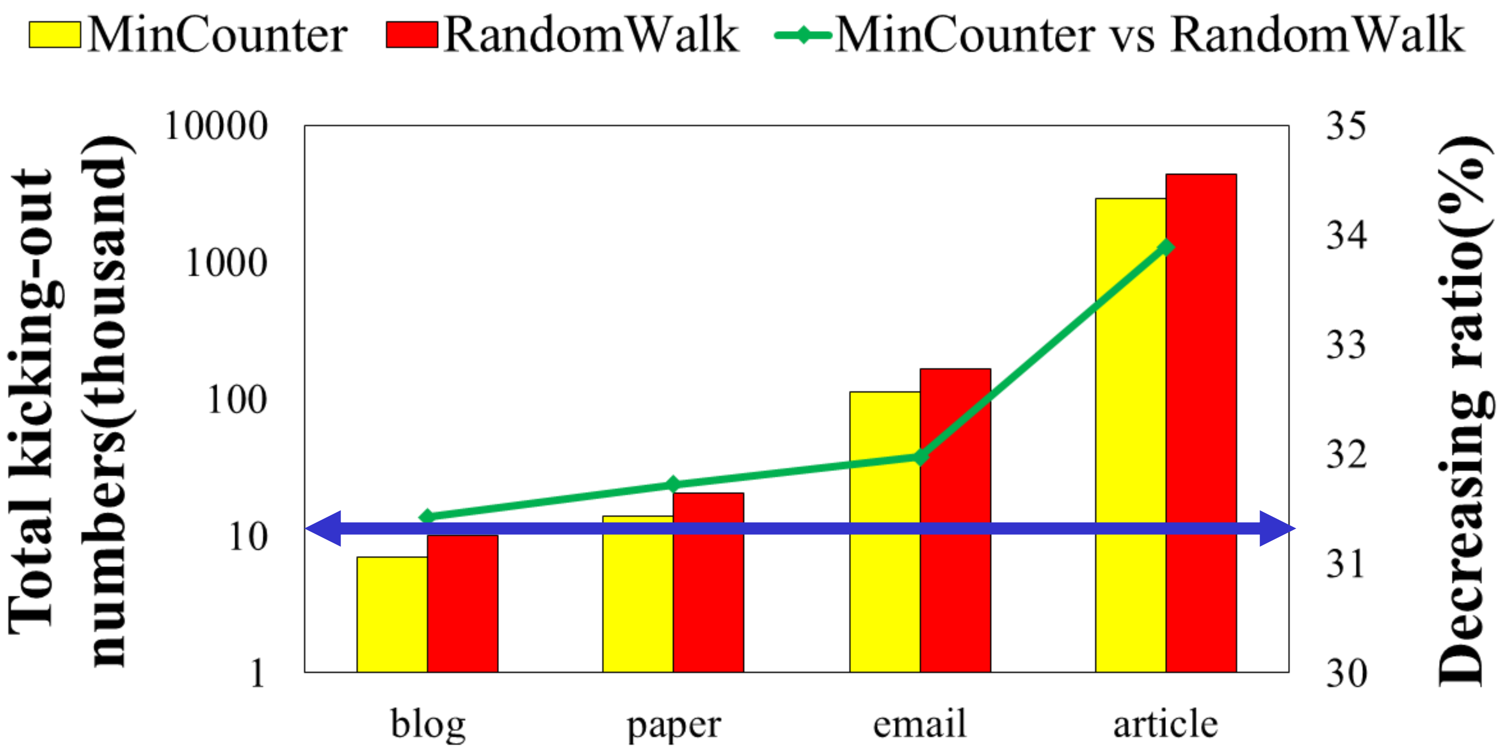
Total kicking-out times(Rate = 1.1)

$$DR = \frac{N\_RandomWalk - N\_MinCounter}{N\_RandomWalk}$$



- MinCounter reduces almost 50% total kicking-out times (R=1.1).

Total kicking-out times(Rate = 2.04)



- MinCounter reduces more than 31% total kicking-out nunbers (R=2.04).

Conclusion

- Endless loops and high insertion latency
- MinCounter selects the “cold” buckets to kick out
  - Alleviate hash collisions
  - Decrease insertion latency
- Substantially decreases the total kicking-out times and improves the utilization ratio of hash tables.
- We release the source code of MinCounter in GitHub.

  https://github.com/syy804123097/MinCounter

Thanks & Questions

Challenges in Cuckoo Hashing

- Intensive kicking out when inserting items
- Endless loops
  - Reconstruct hash tables