Hypersistent Sketch: Enhanced Persistence Estimation via Fast Item Separation

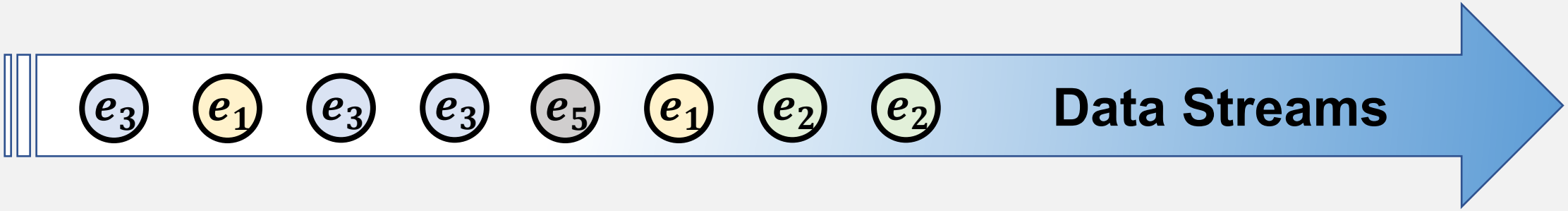
Lu Cao*§‡, Qilong Shi*†, Weiqiang Xiao§, Nianfu Wang§‡, Wenjun Li‡, Zhijun Li§‡, Weizhe Zhang§‡, Mingwei Xu†

§·Harbin Institute of Technology (Shenzhen)

†·Tsinghua University, ‡·Peng Cheng Laboratory

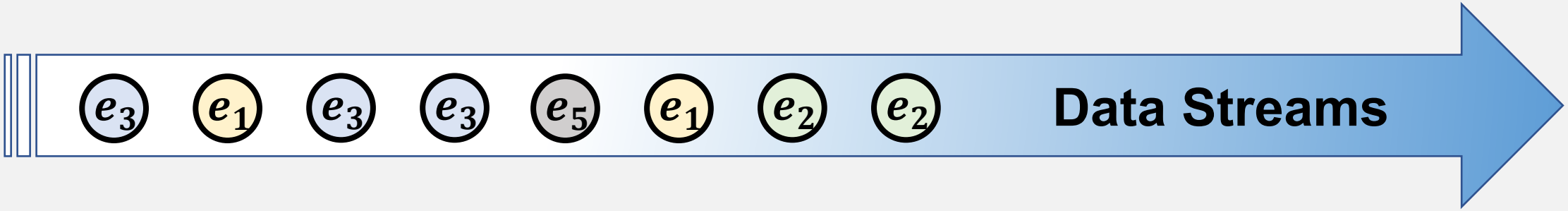
*·*Lu Cao and Qilong Shi are co-first authors of this paper*

Background



High Speed & Large Volume & Skewness v.s. Limited Space

Background

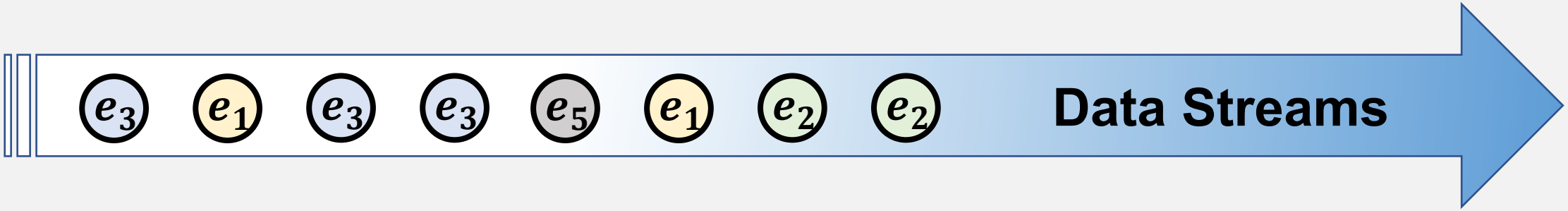


High Speed & Large Volume & Skewness **v.s.** **Limited Space**

Approximate

Persistence, Frequency, Cardinality, Quantile

Background



High Speed & Large Volume & Skewness **v.s.** **Limited Space**

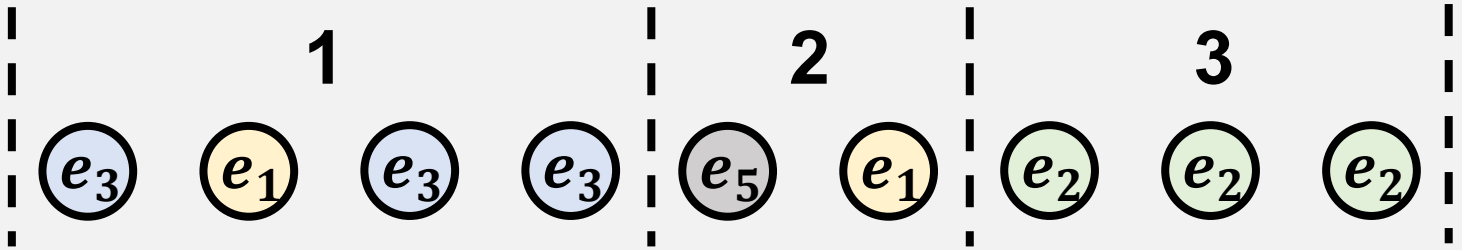
Approximate

Persistence, Frequency, Cardinality, Quantile

Sketch

Persistence

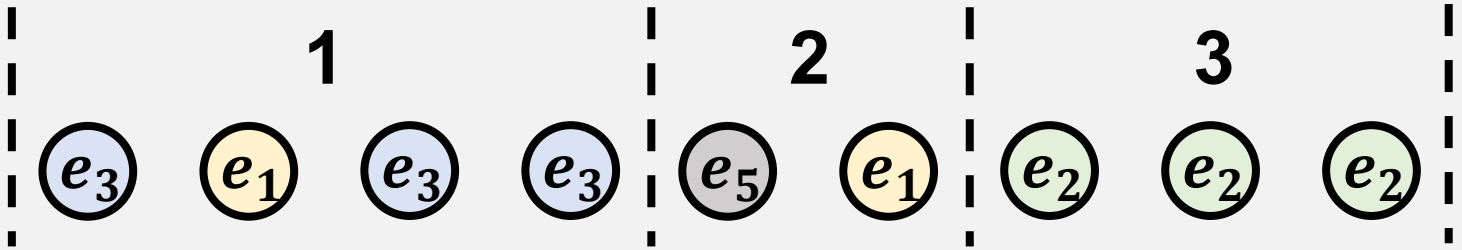
Persistence ≠ Frequency — requires multi-window tracking



Item	Persistence
e_1	2
e_2	1
e_3	1
e_5	1

Persistence

Persistence ≠ Frequency — requires multi-window tracking



Item	Persistence	Frequency
e_1	2	2
e_2	1	3
e_3	1	3
e_5	1	1

Persistence

Challenge

- Balancing memory usage and accuracy in skewed data streams

Problem

- Inefficient memory use and accuracy in persistence estimation and finding persistent items

Two Tasks

Persistence Estimation

- Given any item, report its estimated persistence

Finding Persistent Items

- Report all items whose estimated persistence is larger than a given threshold

Existing Solutions

Persistence Estimation

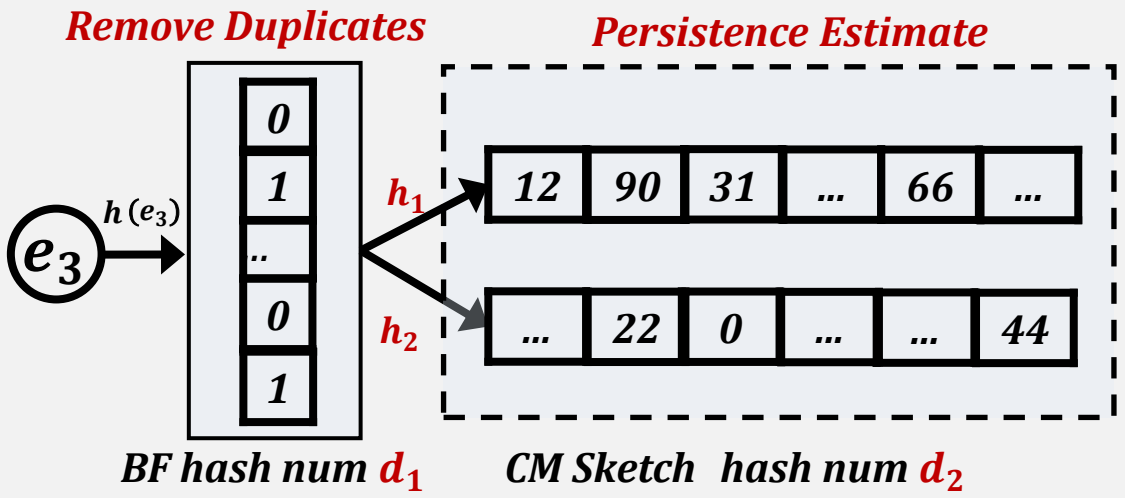
- On-Off Sketch

Finding Persistent Items

- Sketch-based: PIE，On-Off Sketch

PIE Overview

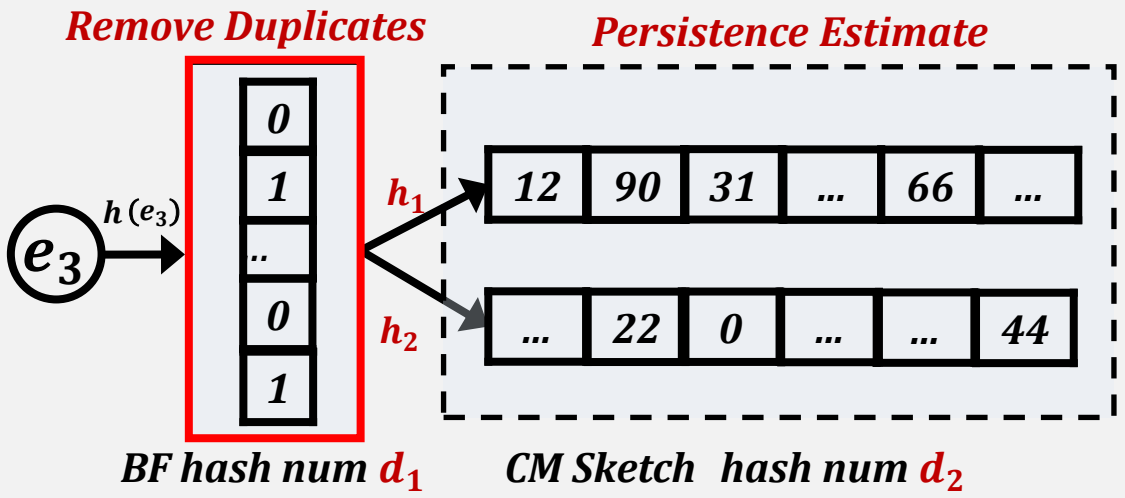
Error = Bloom filter + CM sketch



PIE Limitations

Error = Bloom filter + CM sketch

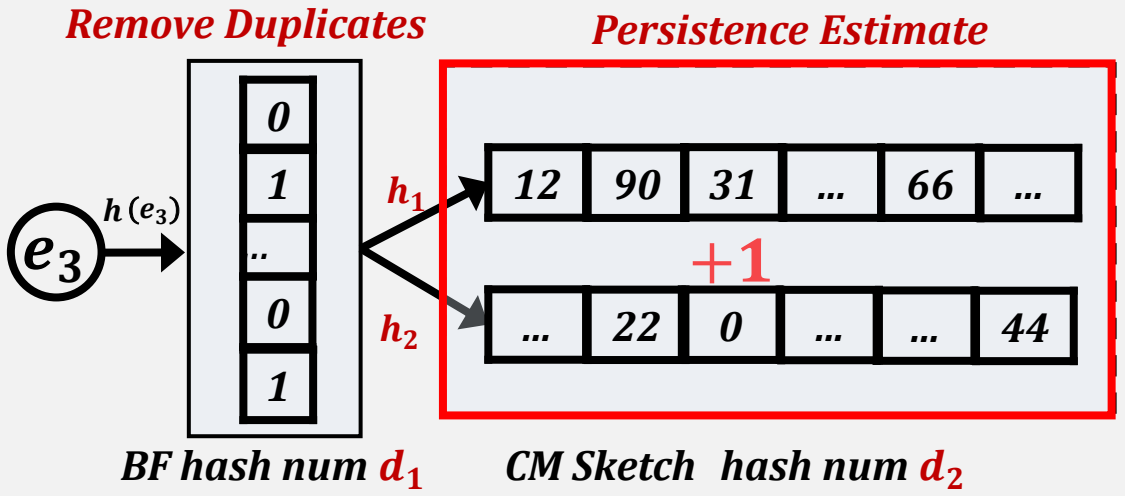
False positive rate



PIE Limitations

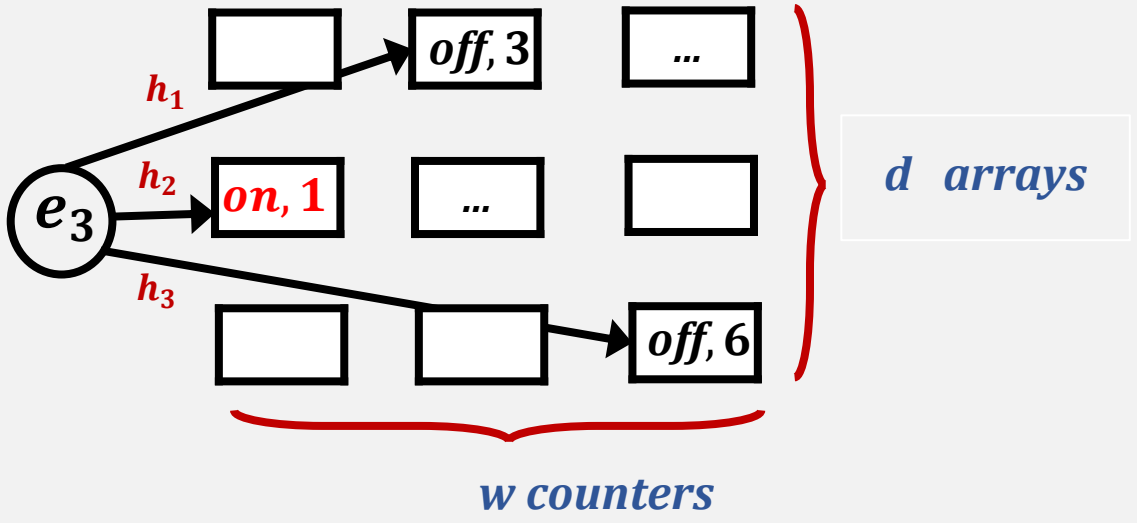
Error = Bloom filter + CM sketch

Hash collisions



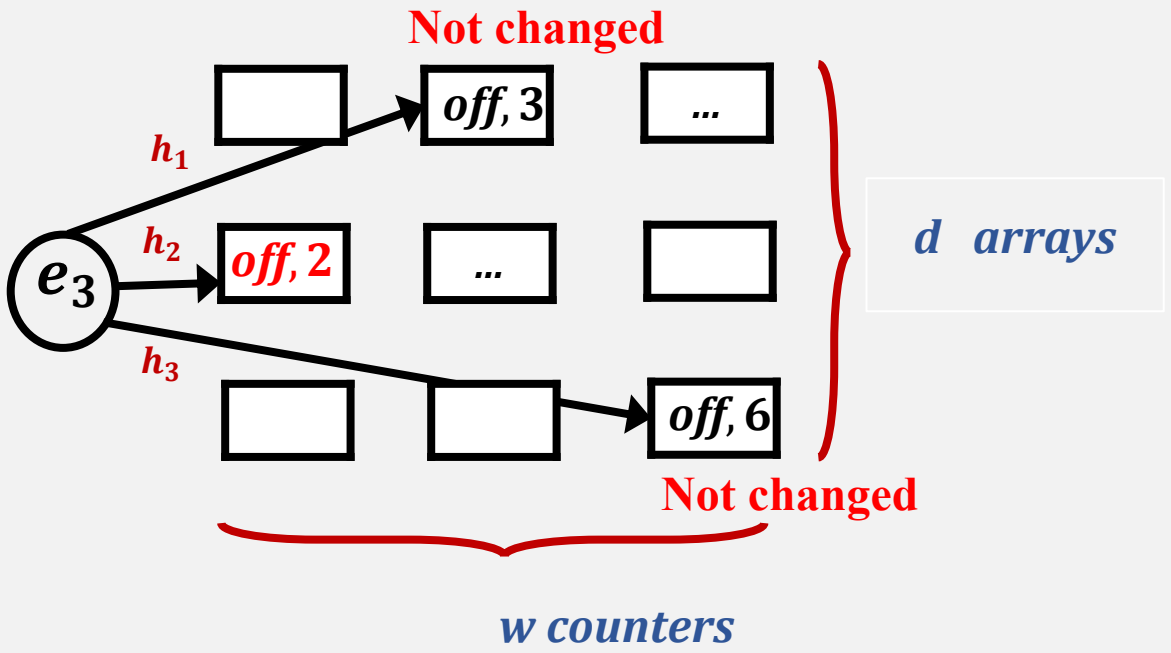
On-Off Sketch Overview

- **Persistence Estimation**
- **Finding Persistent Items**



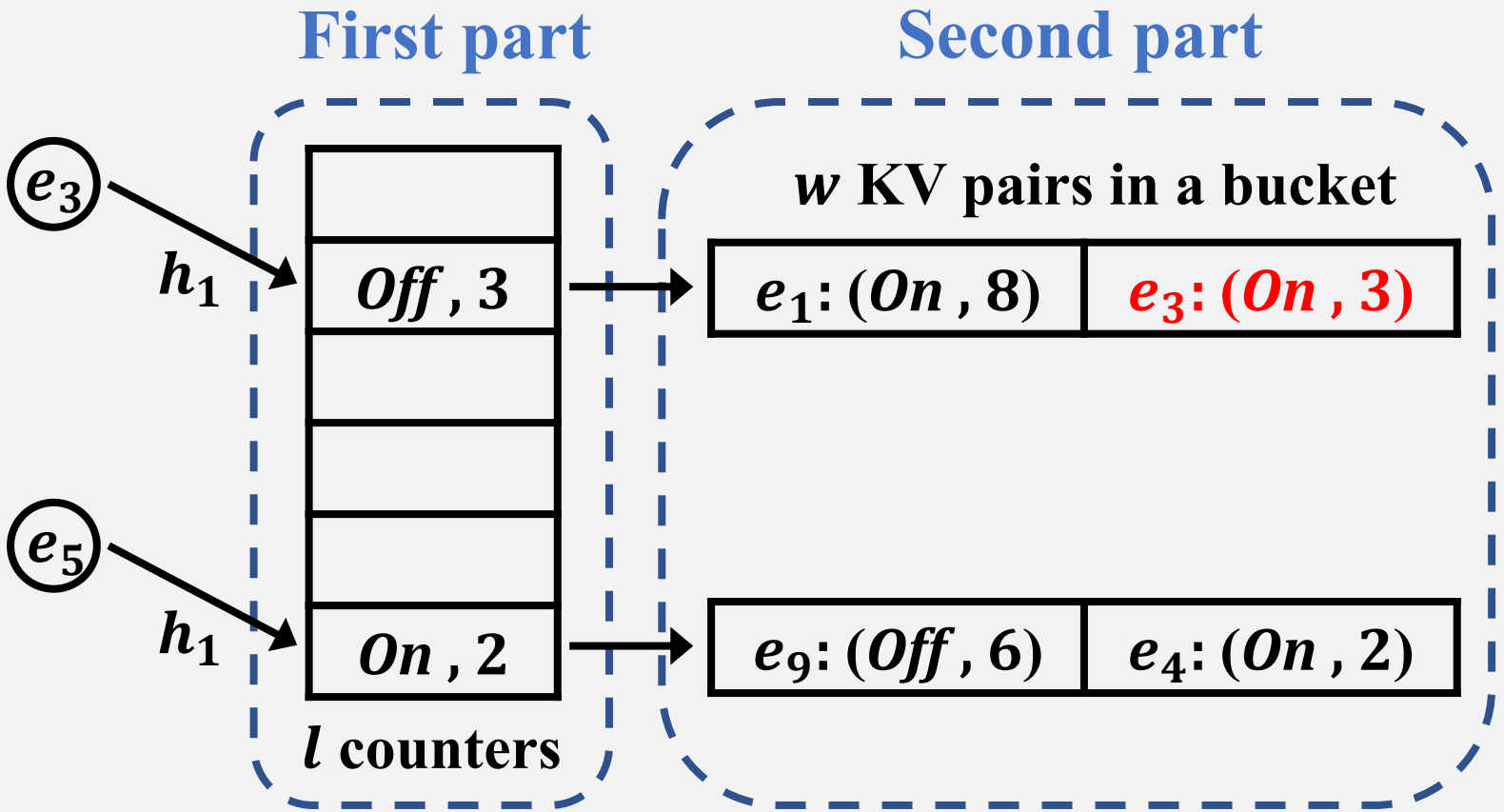
On-Off Sketch Version1

- With on/off flags to prevent multiple increments per window



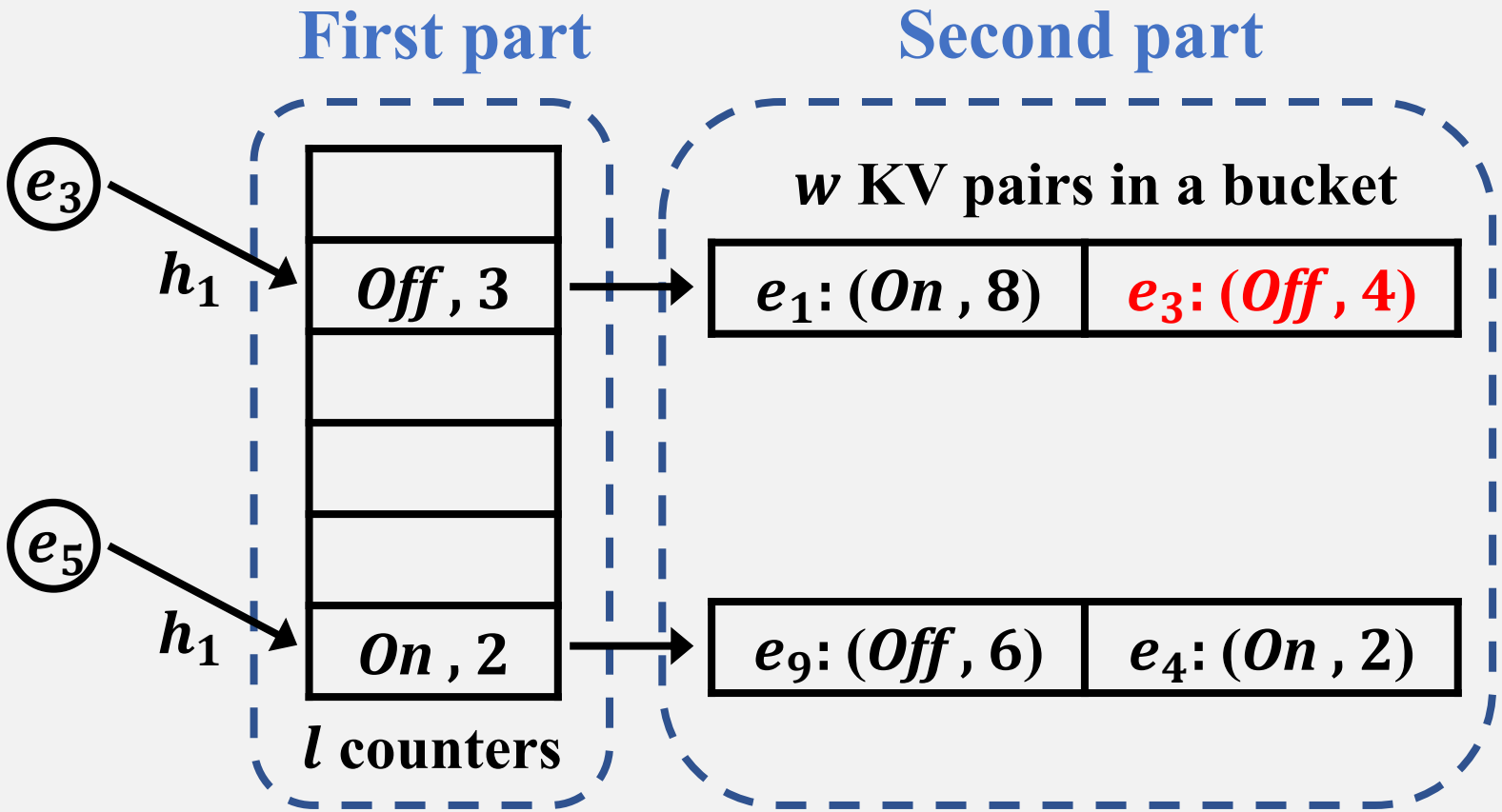
On-Off Sketch Version2

- Persistence Estimation
- Finding Persistent Items



On-Off Sketch Version2

- Bucket-based with item IDs



Limitations of Existing Solutions

- **PIE：Struggles with skewed data**
- **OO Version 1：Skewed data (Few hot items, many cold items)**
- **OO Version 2：Overestimates due to frequent swaps**

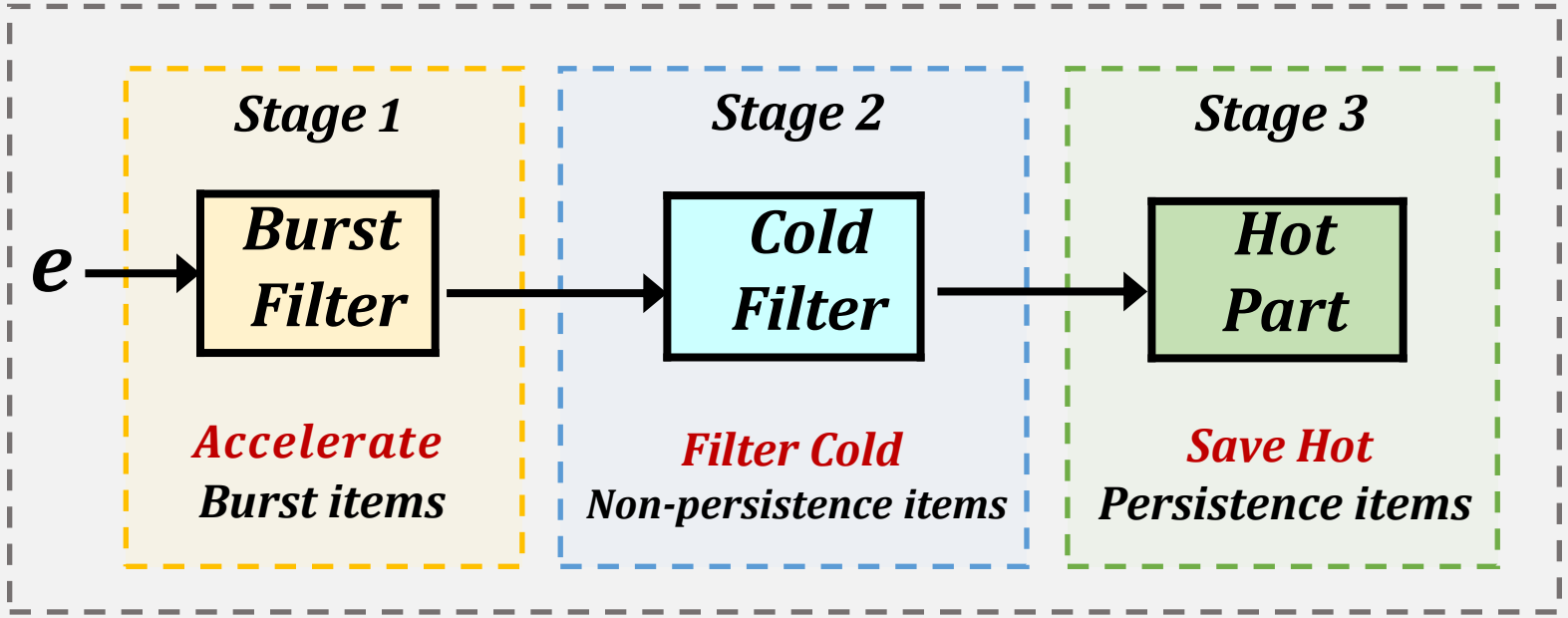
Limitations of Existing Solutions

- **PIE：** Struggles with skewed data
- **OO Version 1：** Skewed data (Few hot items, many cold items)
- **OO Version 2：** Overestimates due to frequent swaps

Motivation

- Skewed data distributions challenge existing solutions.
- Need for efficient handling of cold and hot items to improve accuracy and memory usage

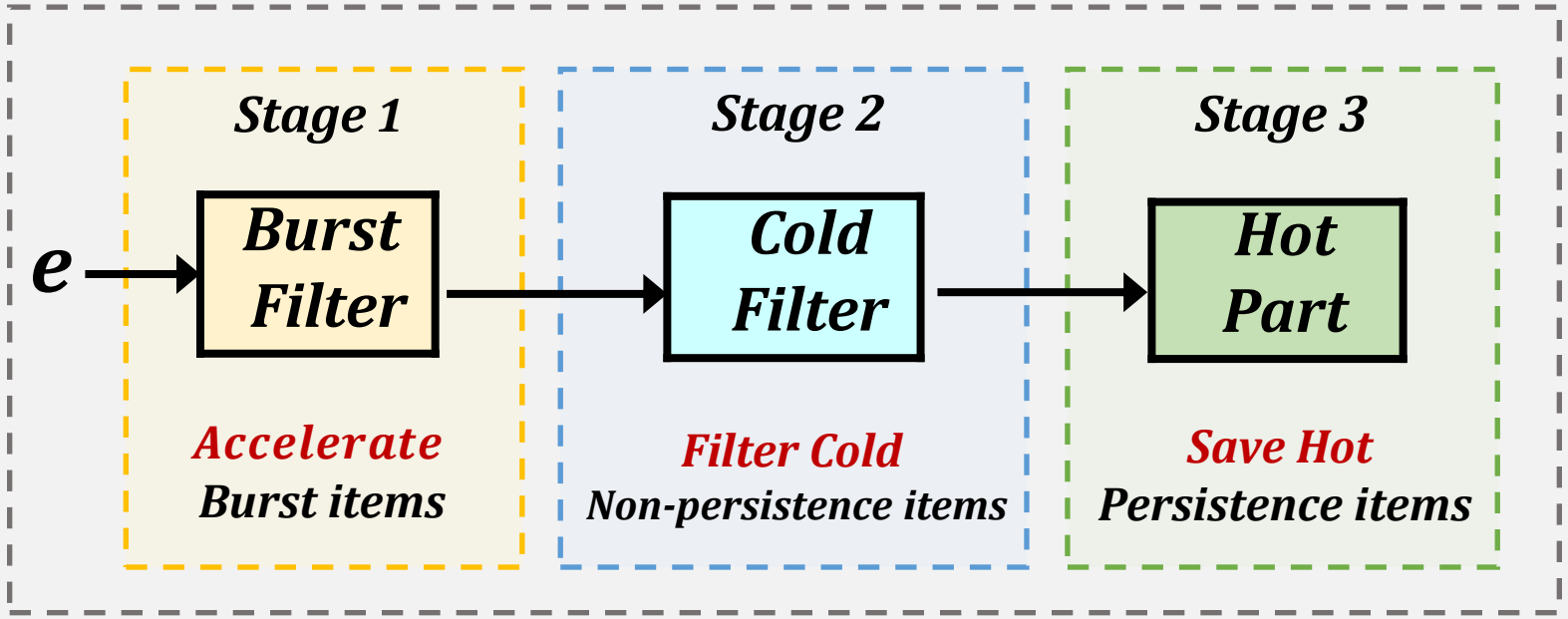
Our Idea



Three-stage design

- Stage 1 Burst Filter (acceleration)
- Stage 2 Cold Filter (cold-items storage)
- Stage 3 Hot Part (hot-items tracking)

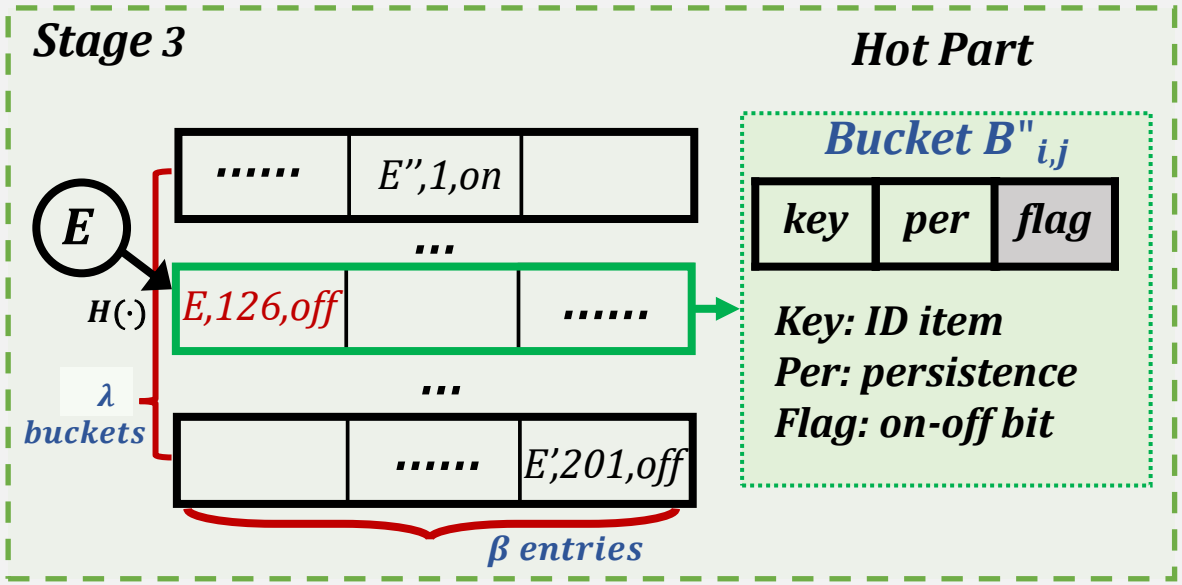
Hypersistent Sketch



Advantages

- Superior accuracy through efficient memory utilization
- Enhanced speed via burst handling mechanism
- Accelerated Persistence Estimation via Hot-Cold Separation

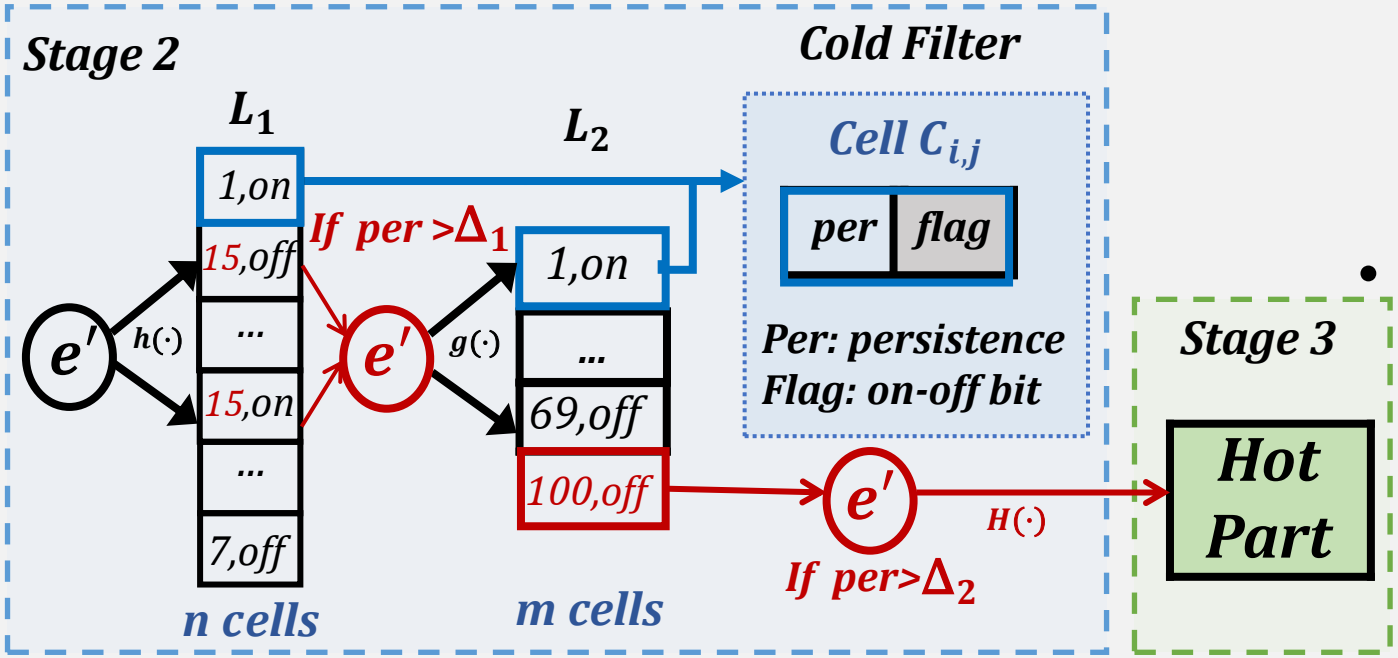
Hypersistent Sketch: Hot Part (Stage 3)



- **Hot Part Structure:**
  - Multi-bucket, Multi-entry Design
- **Update Mechanism:**
  - Single Update per Window via On-Off Flag
- **Replacement Strategy:**
  - Probabilistic Eviction of Least Persistent Item

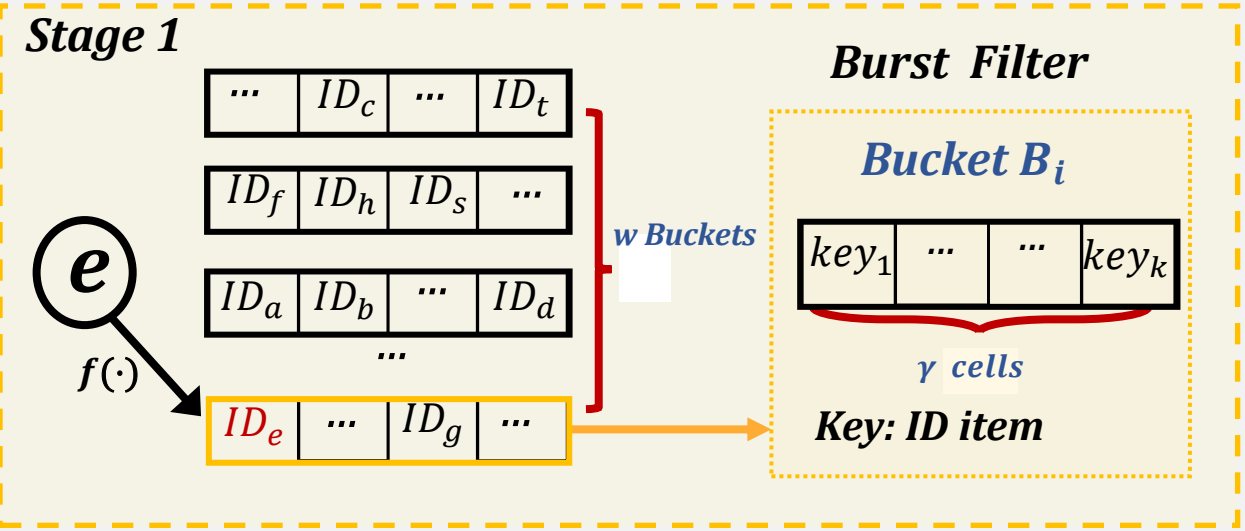
Hypersistent Sketch: Cold Filter (Stage 2)

Problem: Skewed persistence → Cold items may evict hot items



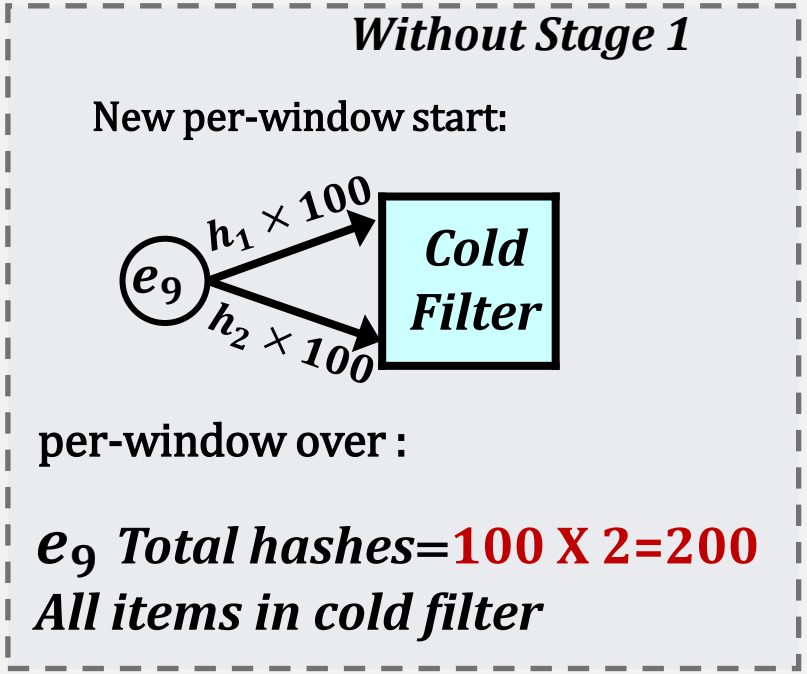
- **Cold Filter Structure:**
  - Two-level filtering structure (L1, L2)
- **Threshold-Based Routing:**
  - Low-persistence items retained in Cold Filter
  - High-persistence items promoted to Hot Part

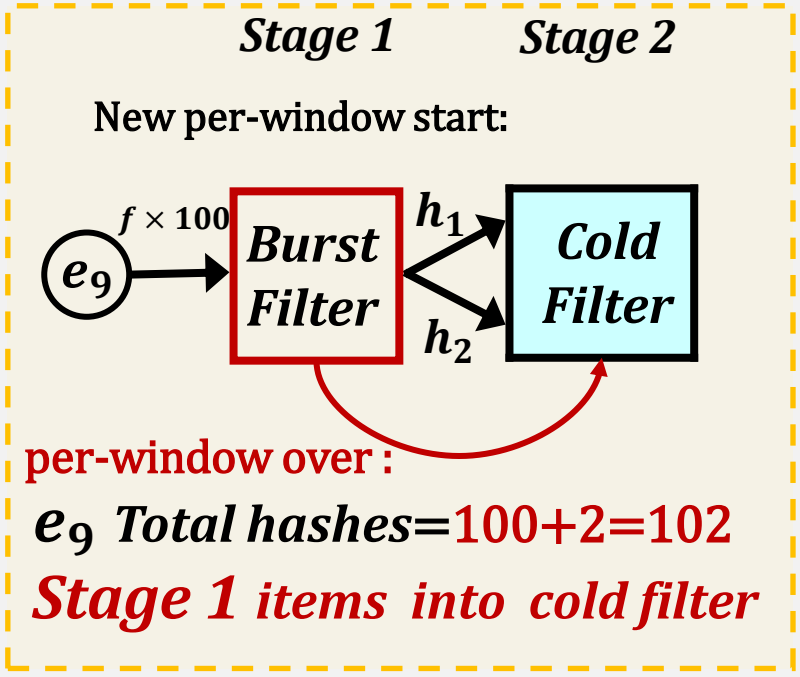
Hypersistent Sketch: Burst Filter (Stage 1)



- **Challenge :**
  - •Cold Filter → Multiple hashes → Throughput
- **Key Insight:**
  - Each cell is updated only once per window
  - Use Burst Filter to record first occurrences only→ Skip redundant hash operations within window
- **Benefit:**
  - Restores throughput while preserving estimation accuracy

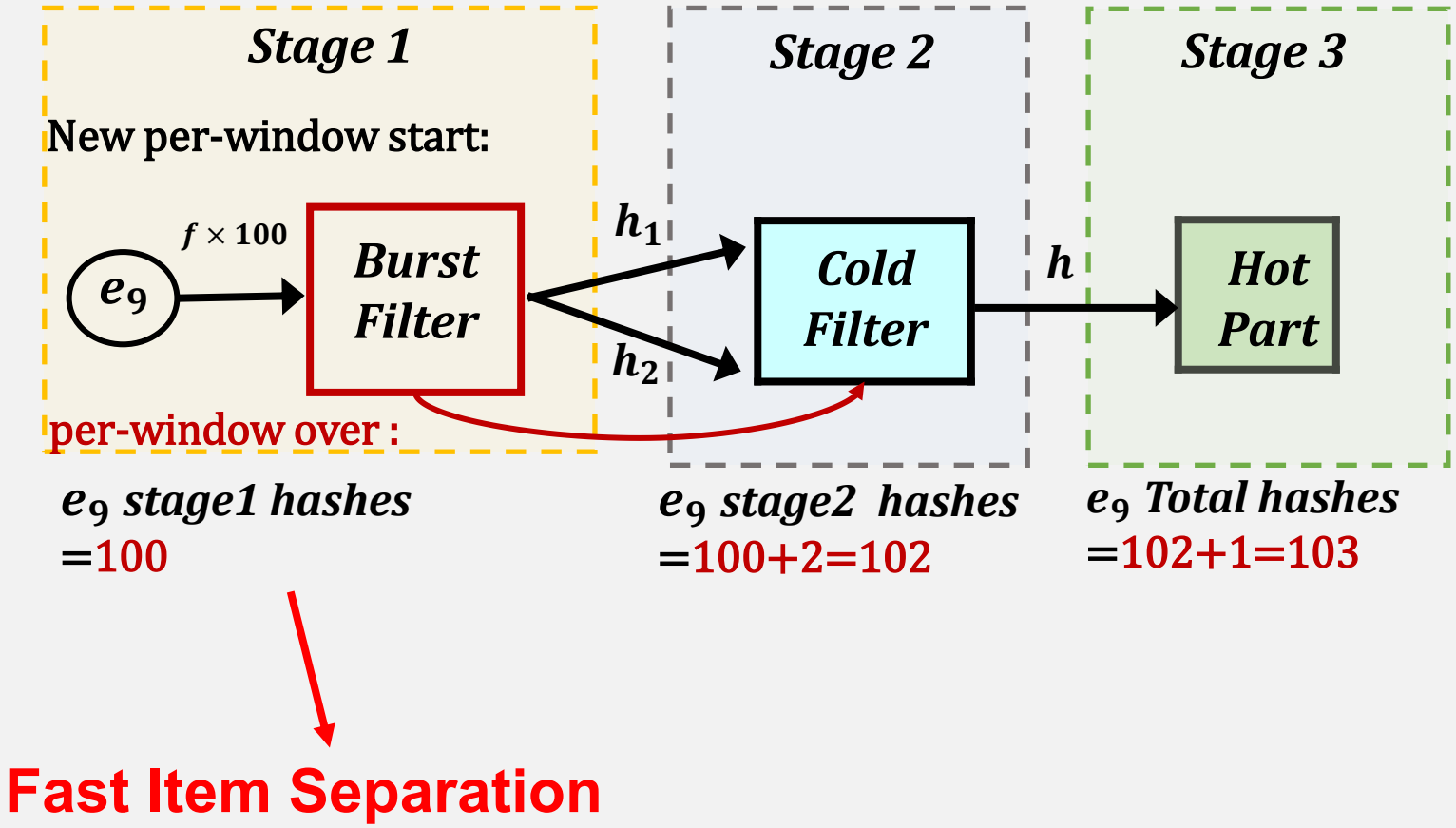
Hypersistent Sketch: Burst Filter (Stage 1)



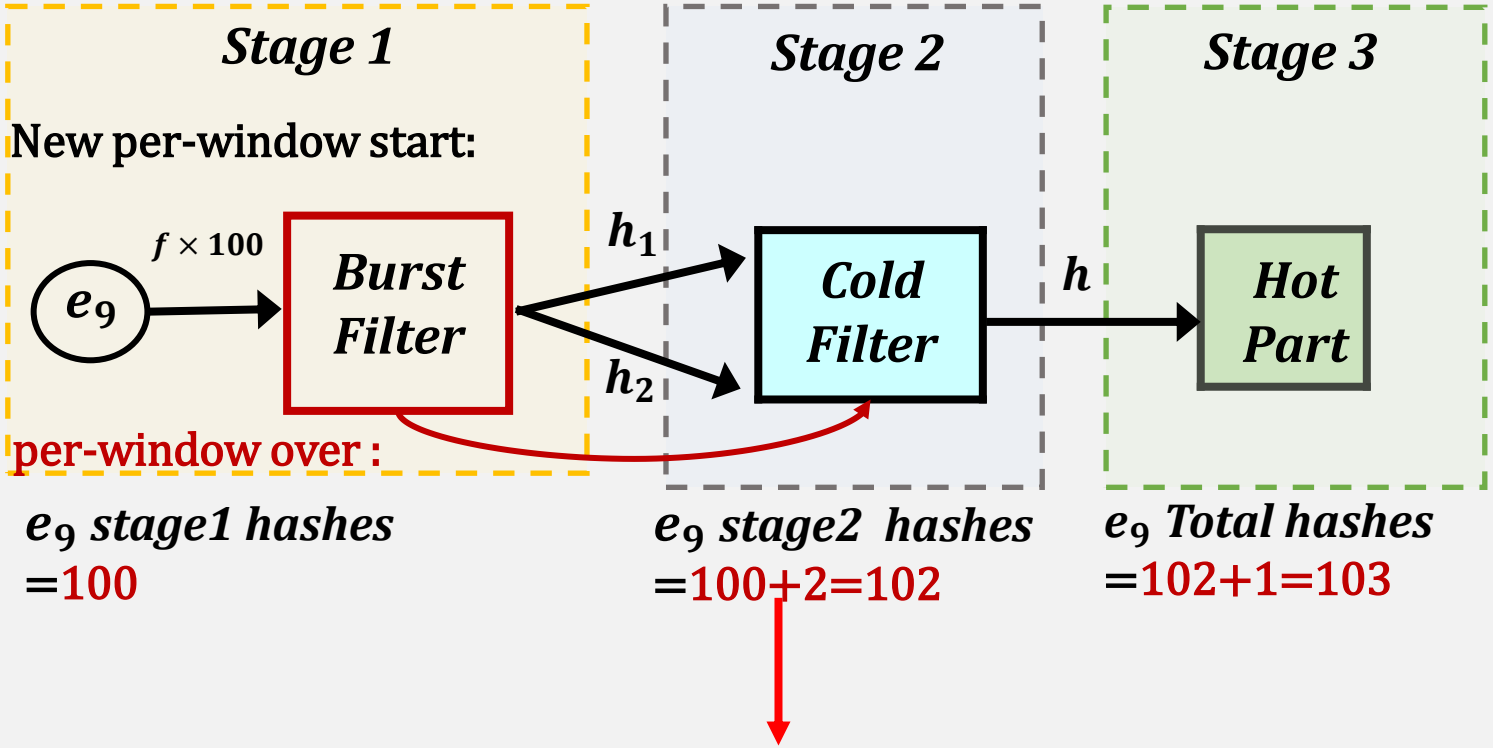


- Reduces Hash Computations
- Saves 98 Computations

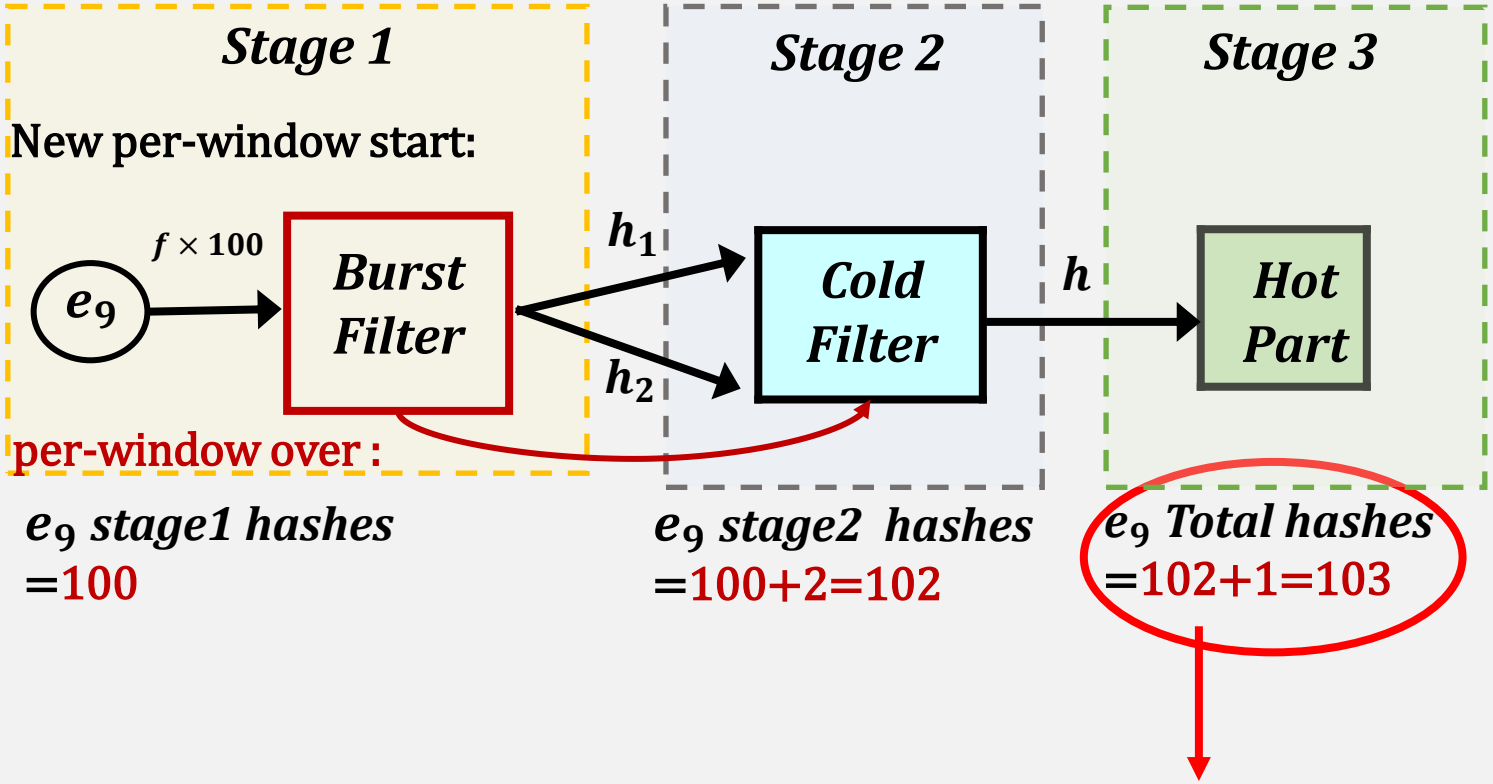
Hypersistent Sketch: Burst Filter (Stage 1)



Hypersistent Sketch: Burst Filter (Stage 1)

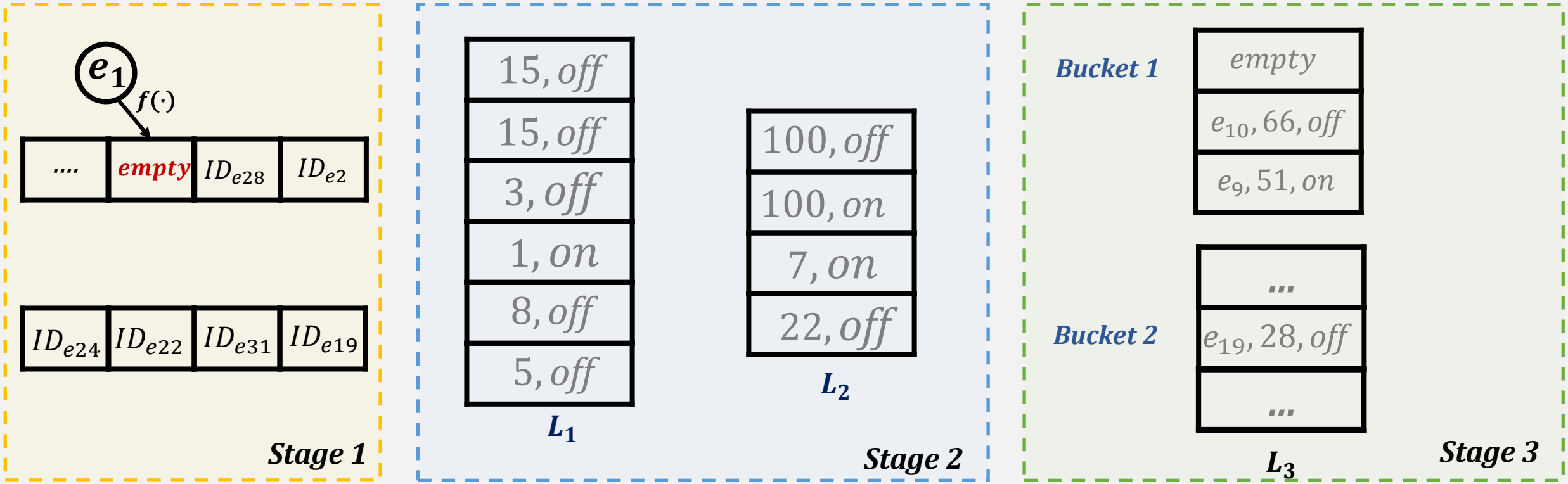


Hypersistent Sketch: Burst Filter (Stage 1)

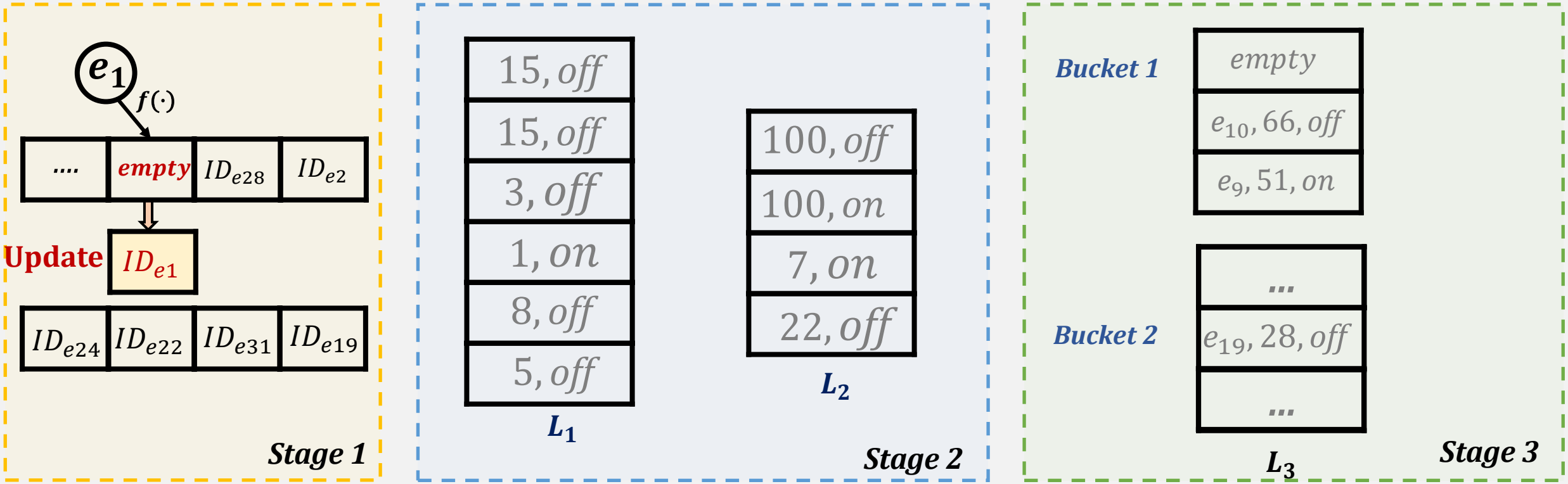


Optimized for Hot-persistent Items

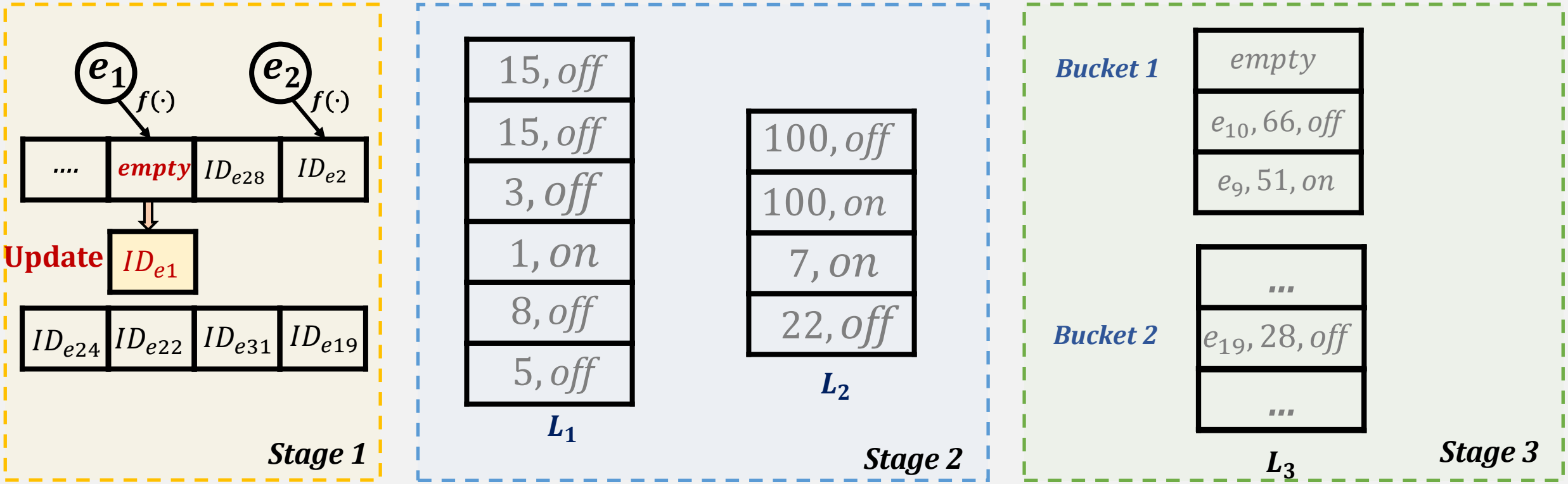
Hypersistent Sketch: A Running Example



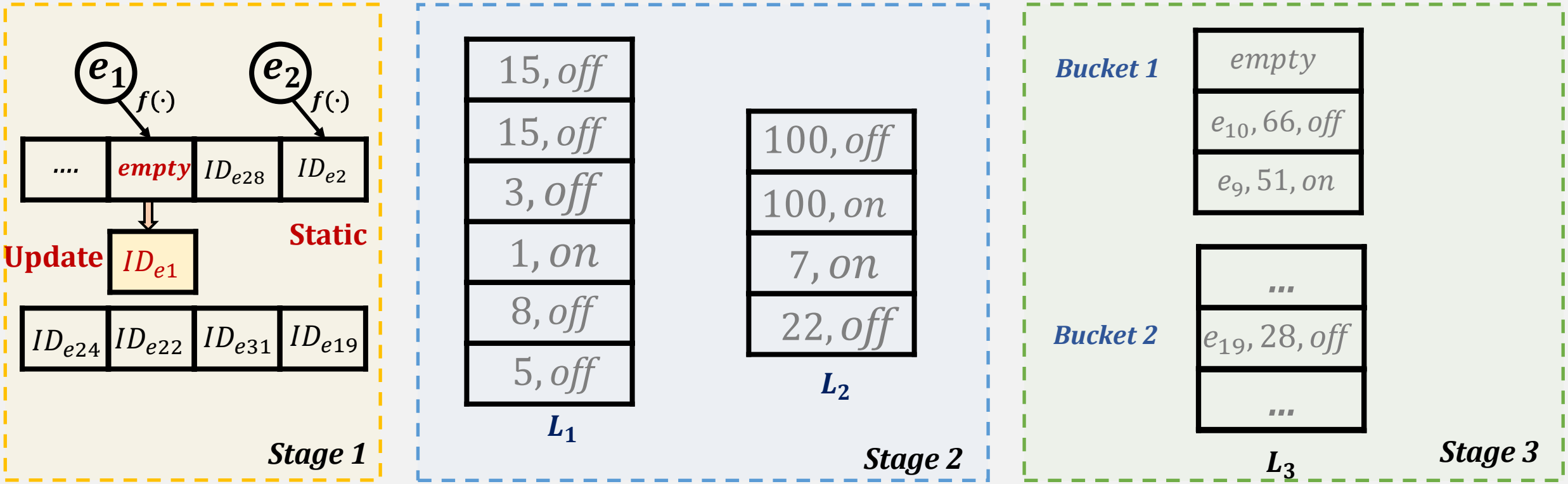
Hypersistent Sketch: A Running Example



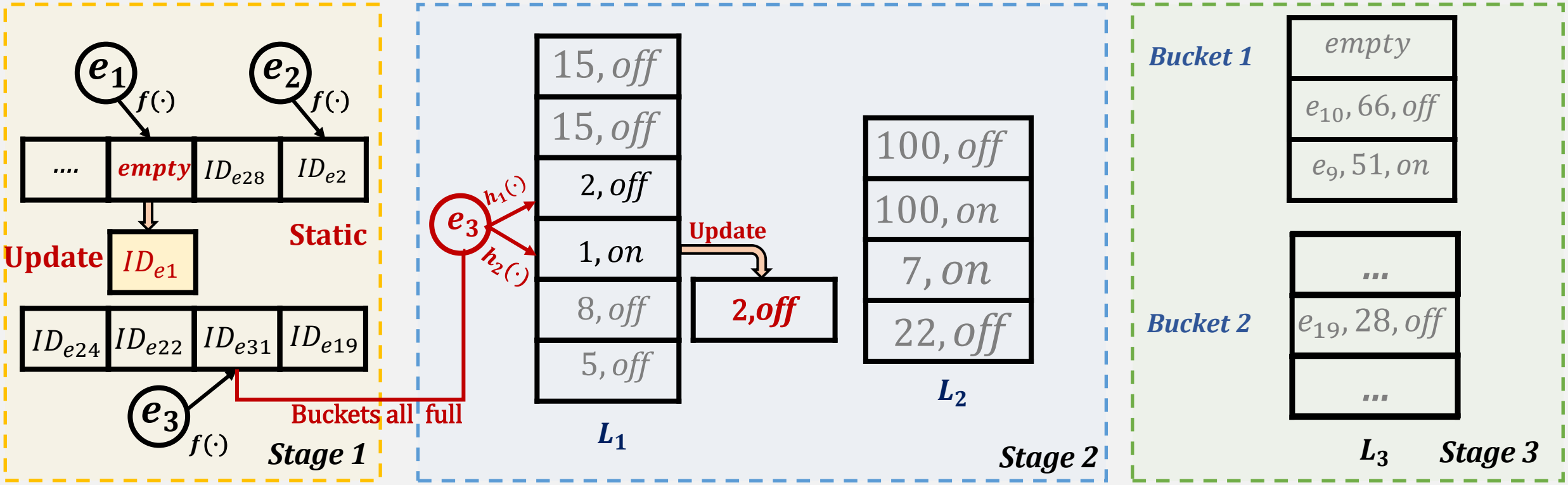
Hypersistent Sketch: A Running Example



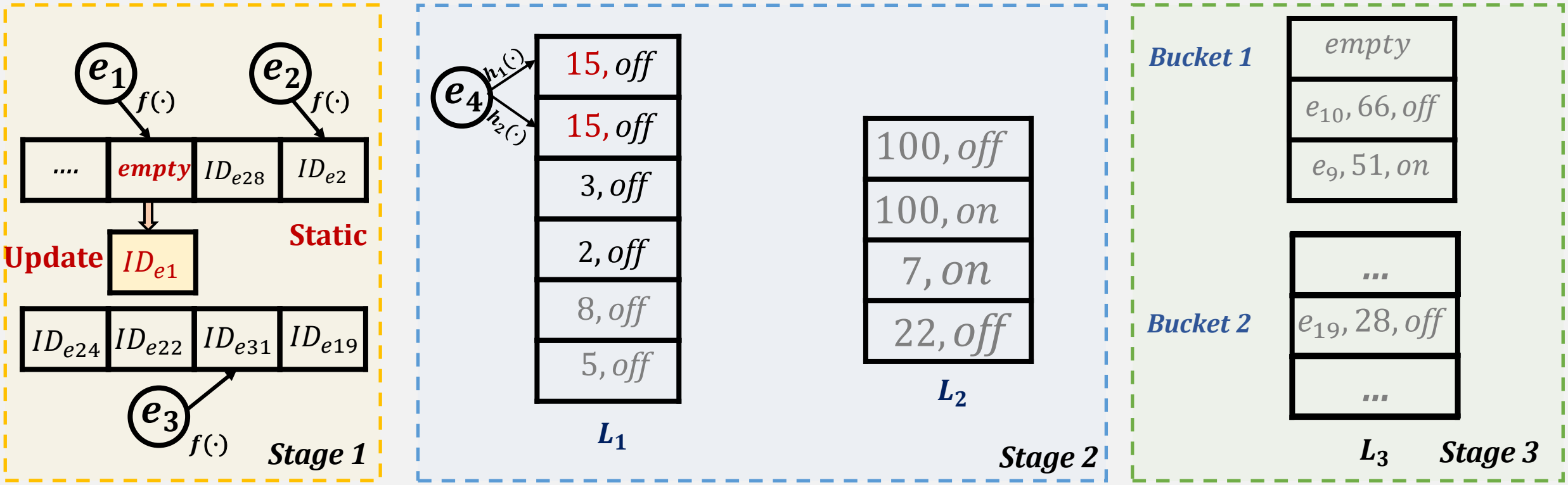
Hypersistent Sketch: A Running Example



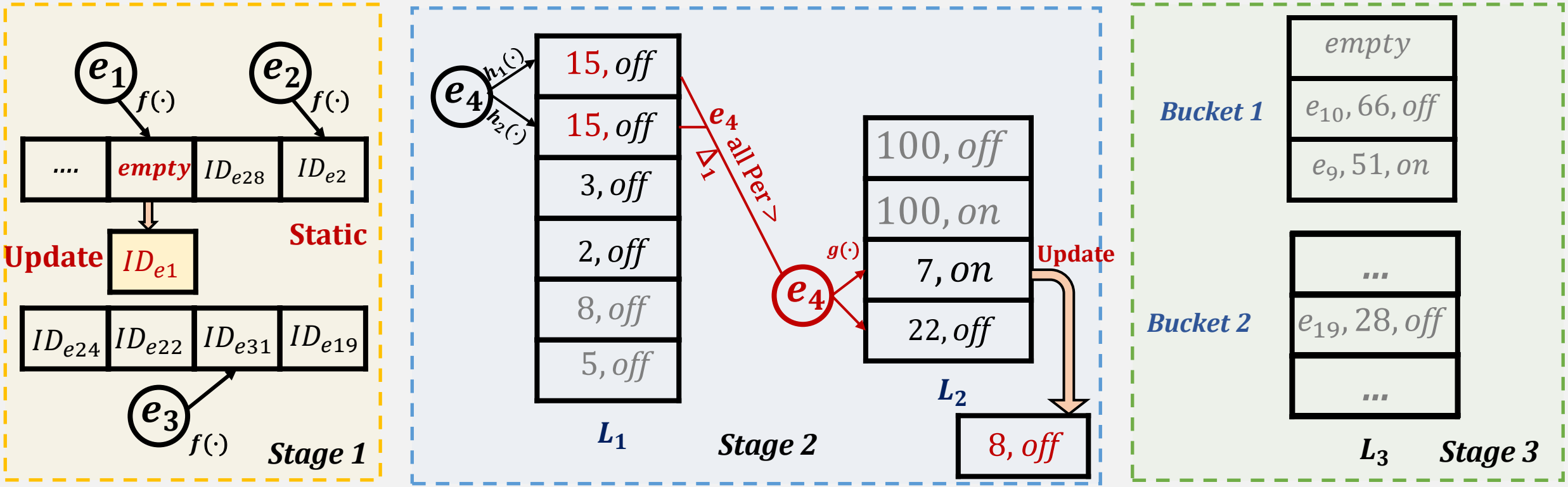
Hypersistent Sketch: A Running Example



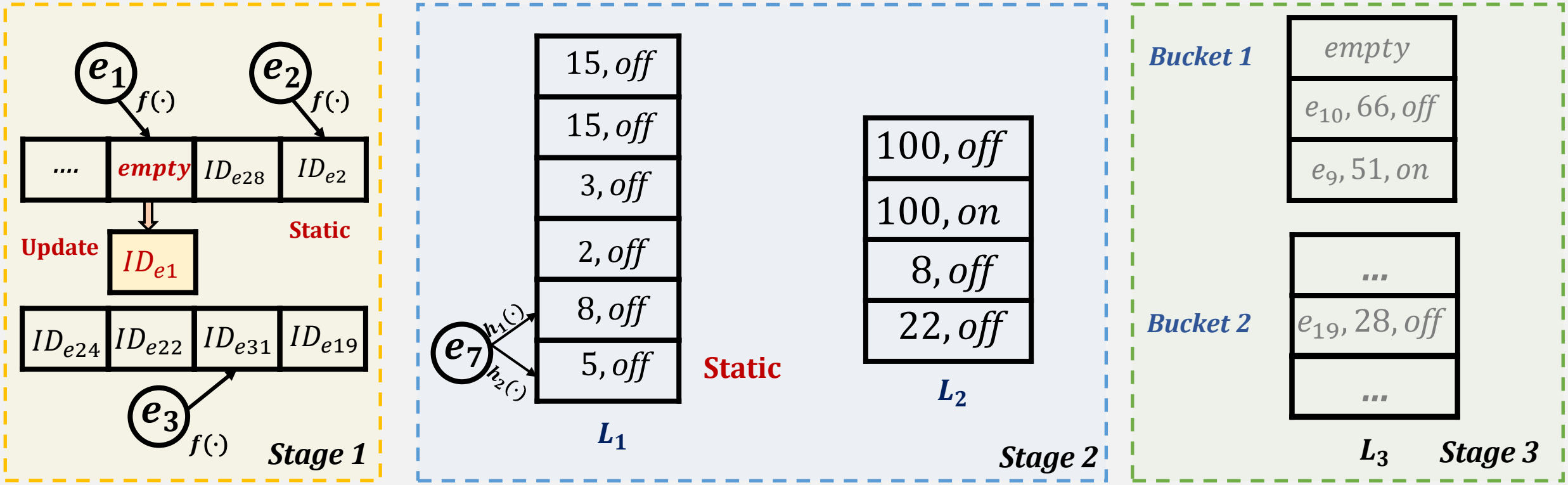
Hypersistent Sketch: A Running Example



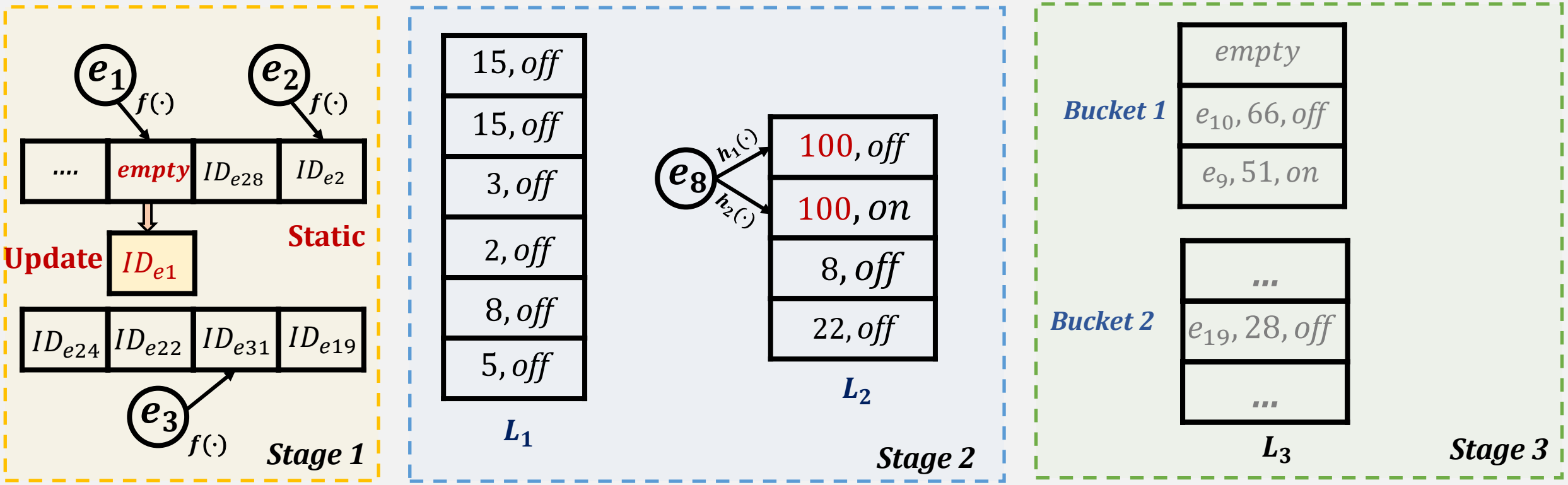
Hypersistent Sketch: A Running Example



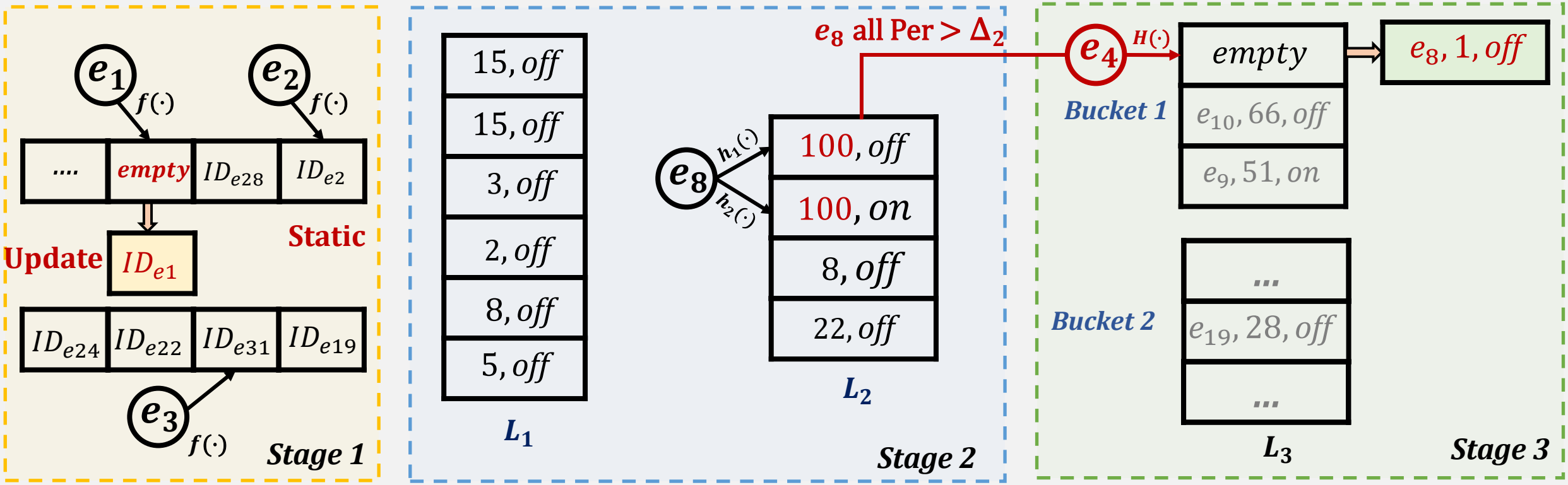
Hypersistent Sketch: A Running Example



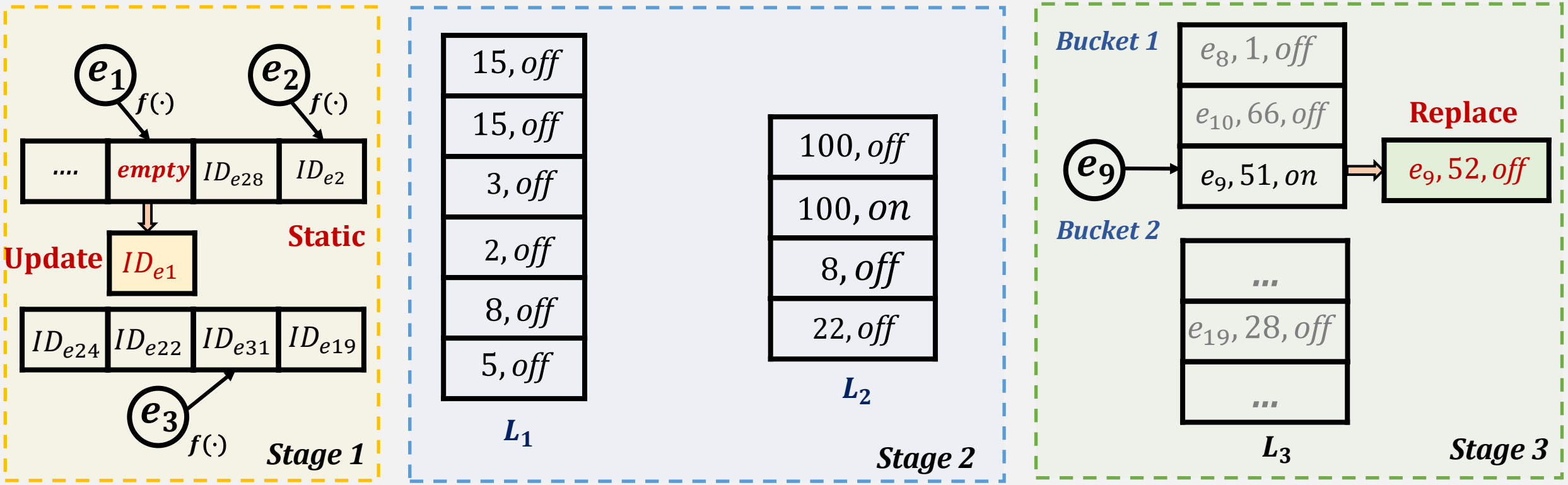
Hypersistent Sketch: A Running Example



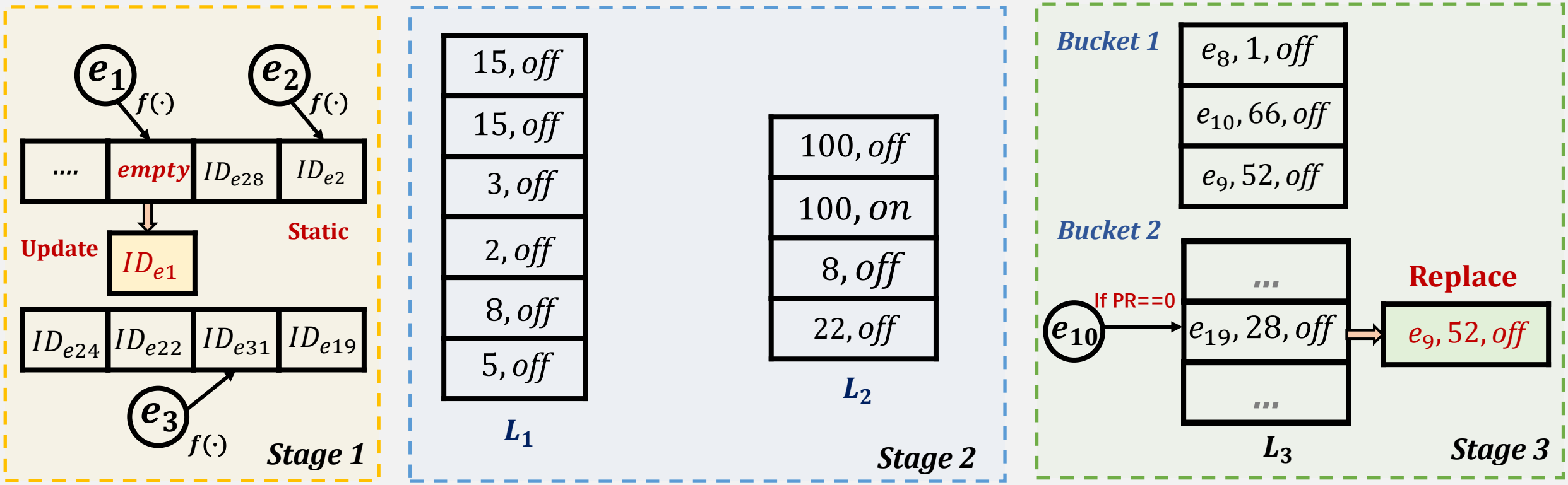
Hypersistent Sketch: A Running Example



Hypersistent Sketch: A Running Example



Hypersistent Sketch: A Running Example



Theoretical Result

Persistence Estimation

- Error-Bounded Estimation
- High-Probability Guarantees
- Lower Estimation Error under Equal Memory

Theoretical Result

Persistence Estimation

- Error-Bounded Estimation
- High-Probability Guarantees
- Lower Estimation Error under Equal Memory

Finding Persistent Items

- Proven Accuracy Ordering
- Cold-Item Filtering Reduces Hash Collision
- Lower Overestimation Compared to On-Off Sketch

Experiments

Datasets

- Synthetic, MAWI, Campus, CAIDA

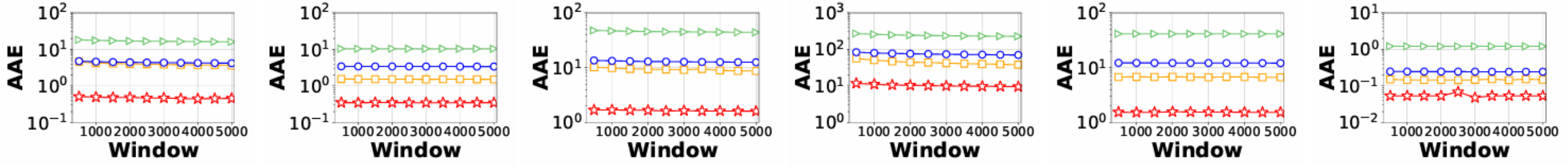
Persistence Estimation Metrics

- AAE *vs. windows*, AAE *vs. windows*, ARE *vs. memory*, ARE *vs. memory*

Finding Persistent Items Metrics

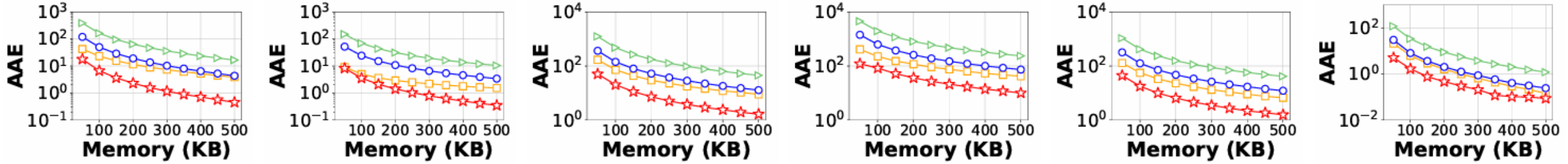
- ARE, FPR, FNR, F1 Score

Experiments: Persistence Estimation



(a) CAIDA (b) MAWI (c) Campus (d) Big CAIDA (e) Zipf 1.5 (f) Zipf 2.0

Fig. 11: AAE on persistence estimation vs. window.

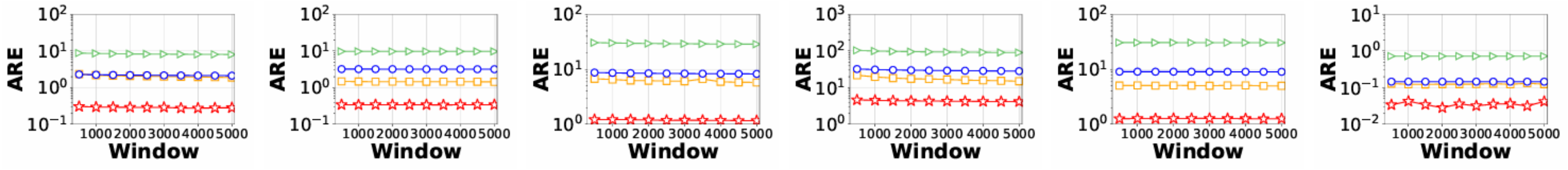


(a) CAIDA (b) MAWI (c) Campus (d) Big CAIDA (e) Zipf 1.5 (f) Zipf 2.0

Fig. 12: AAE on persistence estimation vs. memory.

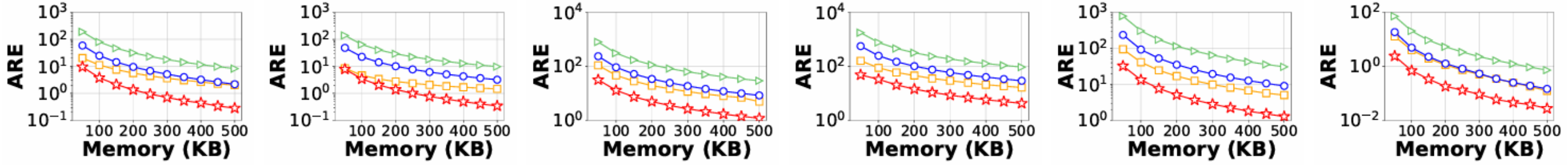
- 👉 Stable Accuracy
- 👉 Error-Resilient

Experiments: Persistence Estimation



(a) CAIDA (b) MAWI (c) Campus (d) Big CAIDA (e) Zipf 1.5 (f) Zipf 2.0

Fig. 13: ARE on persistence estimation vs. window.

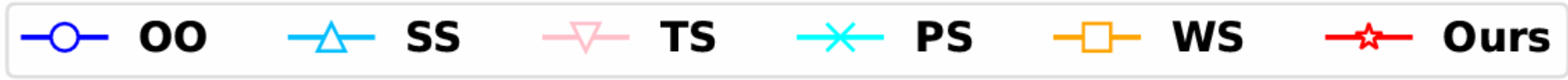


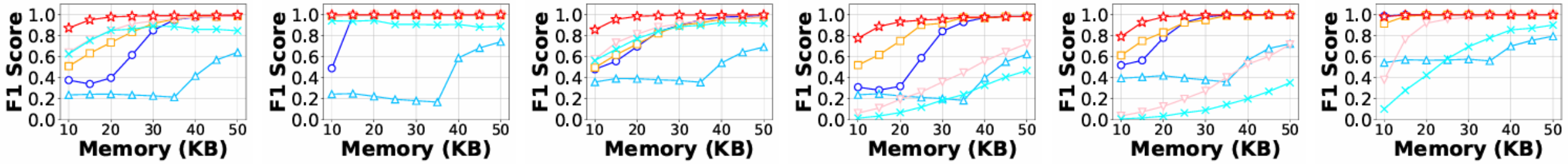
(a) CAIDA (b) MAWI (c) Campus (d) Big CAIDA (e) Zipf 1.5 (f) Zipf 2.0

Fig. 14: ARE on persistence estimation vs. memory.

- 👉 Memory-Efficient
- 👉 Friendly to persistent items

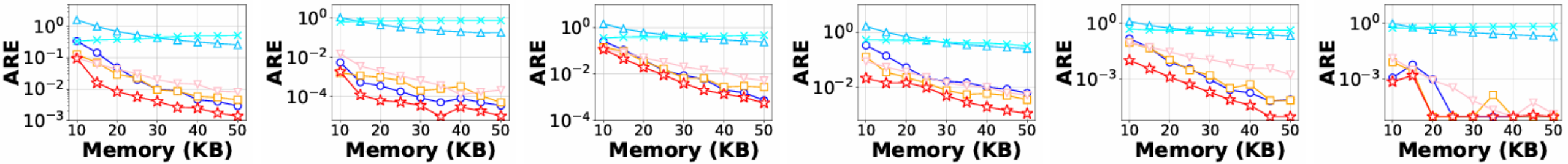
Experiments: Finding Persistent Items





(a) CAIDA (b) MAWI (c) Campus (d) Big CAIDA (e) Zipf 1.5 (f) Zipf 2.0

Fig. 15: F1-score on finding persistent items.

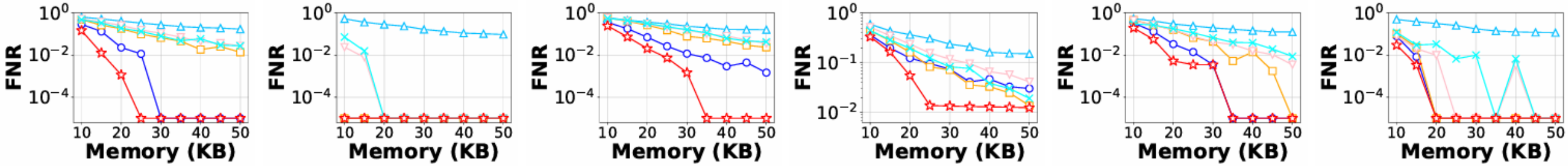


(a) CAIDA (b) MAWI (c) Campus (d) Big CAIDA (e) Zipf 1.5 (f) Zipf 2.0

Fig. 16: ARE on finding persistent items.

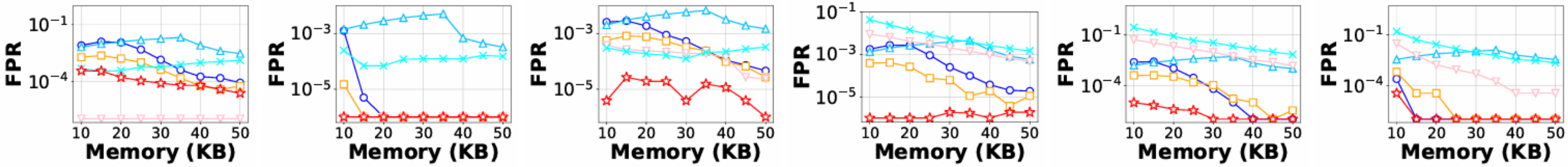
- 👉 F1 ↑ Best Overall Balance
- 👉 Average Relative Error (ARE) ↓ Lowest Across All

Experiments: Finding Persistent Items



(a) CAIDA (b) MAWI (c) Campus (d) Big CAIDA (e) Zipf 1.5 (f) Zipf 2.0

Fig. 17: FNR on finding persistent items.



(a) CAIDA (b) MAWI (c) Campus (d) Big CAIDA (e) Zipf 1.5 (f) Zipf 2.0

Fig. 18: FPR on finding persistent items.

- 👉 False Positive Rates (FPR) ↓ Orders of Magnitude Lower
- 👉 False Negative Rates (FNR) ↓ Strong in Detecting Positives

Experiments: SIMD

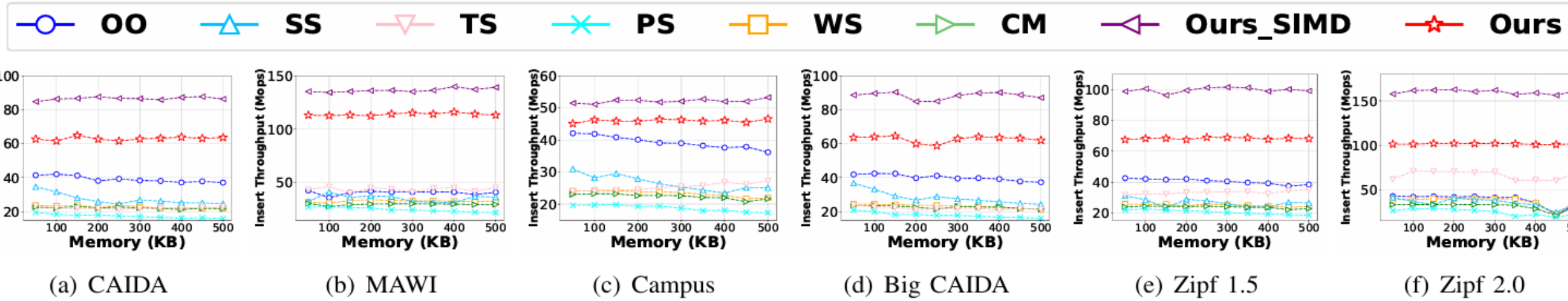


Fig. 19: Insert Throughput of Hypersistent Sketch with/without SIMD.

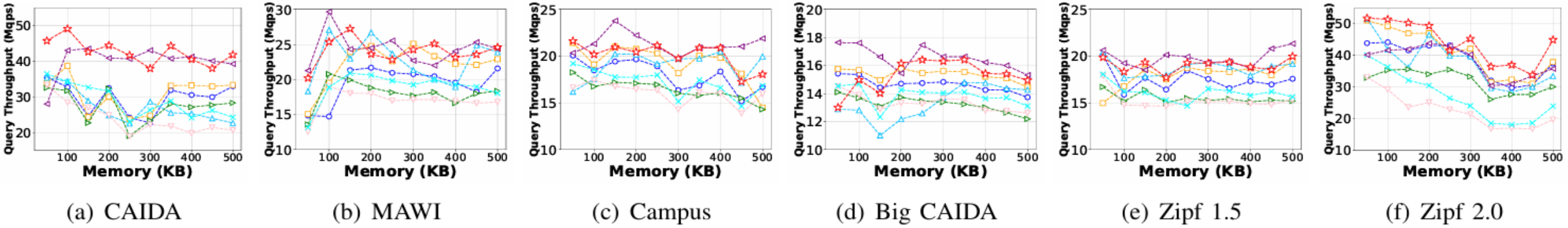


Fig. 20: Query Throughput of Hypersistent Sketch with/without SIMD.

- Single Instruction Multiple Data（SIMD） Acceleration

👉 50% Faster Throughput

Summary

- **Key Takeaways**
  - ✔ **Two tasks on persistence**
  - ✔ **Burst Filter** avoids redundant updates
  - ✔ **Cold Filter** removes low-persistence items
  - ✔ **Hot Part** ensures precise tracking of high-value item
- **Outperforms On-Off Sketch with**
  - ✔ Higher accuracy
  - ✔ **50%** Faster Throughput
  - ✔ Lower memory usage

Thank you!

Our code is open-source!

https://github.com/wenjunpaper/HypersistentSketch

stallone@pku.edu.cn
wenjunli@pku.org.cn