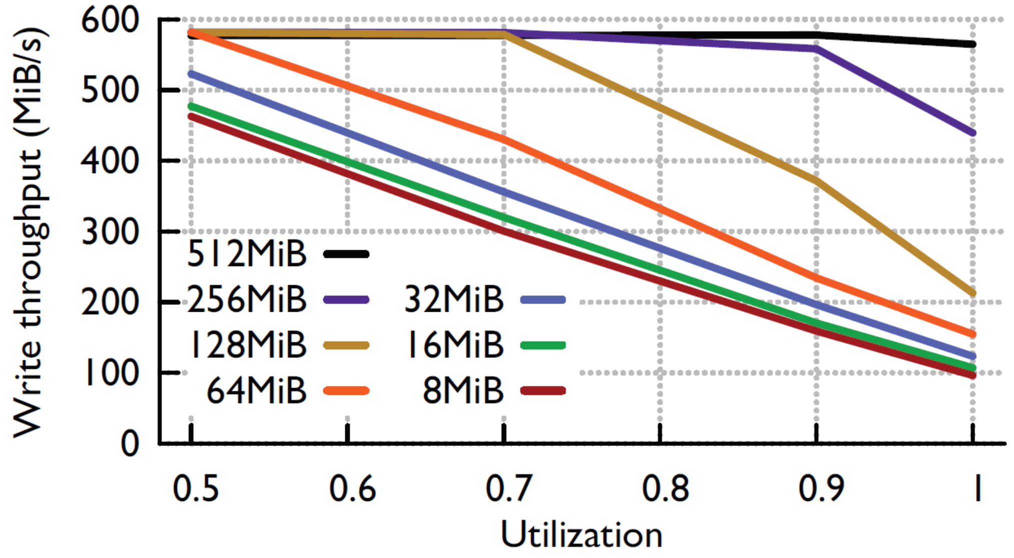
RIPQ

Advanced Photo Caching on Flash for Facebook

Linpeng Tang, Qi Huang, Wyatt Lloyd, Sanjeev Kumar, Kai Li

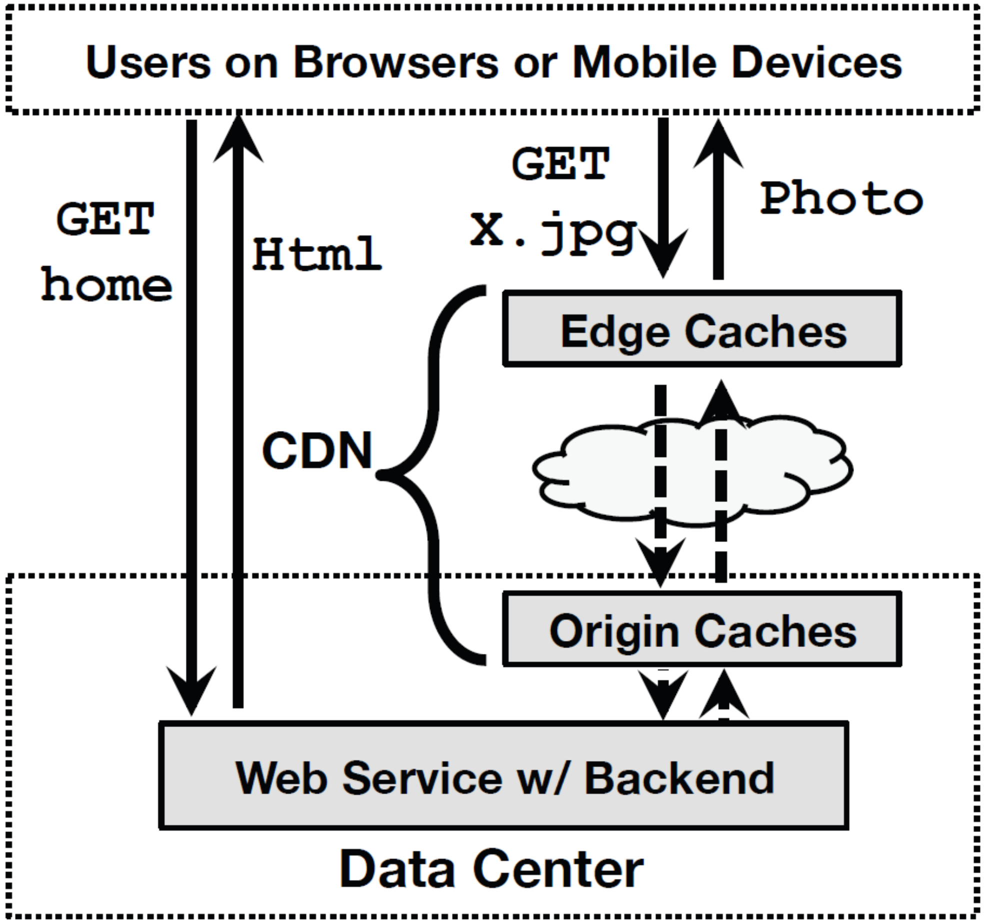
Motivation

- Need to improve **hit ratios** in Facebook's **flash-based** distributed photo caches
- Advanced caching algorithms generate many **small random writes**
  - Frequent garbage collection in FTL ⇒ *high write amplification, shorter device lifespan*
  - Small writes ⇒ *low throughput*
- Result: Facebook used FIFO caches
  - Frequent cache misses ⇒ *many back-end I/Os*



Background: Cache Layers

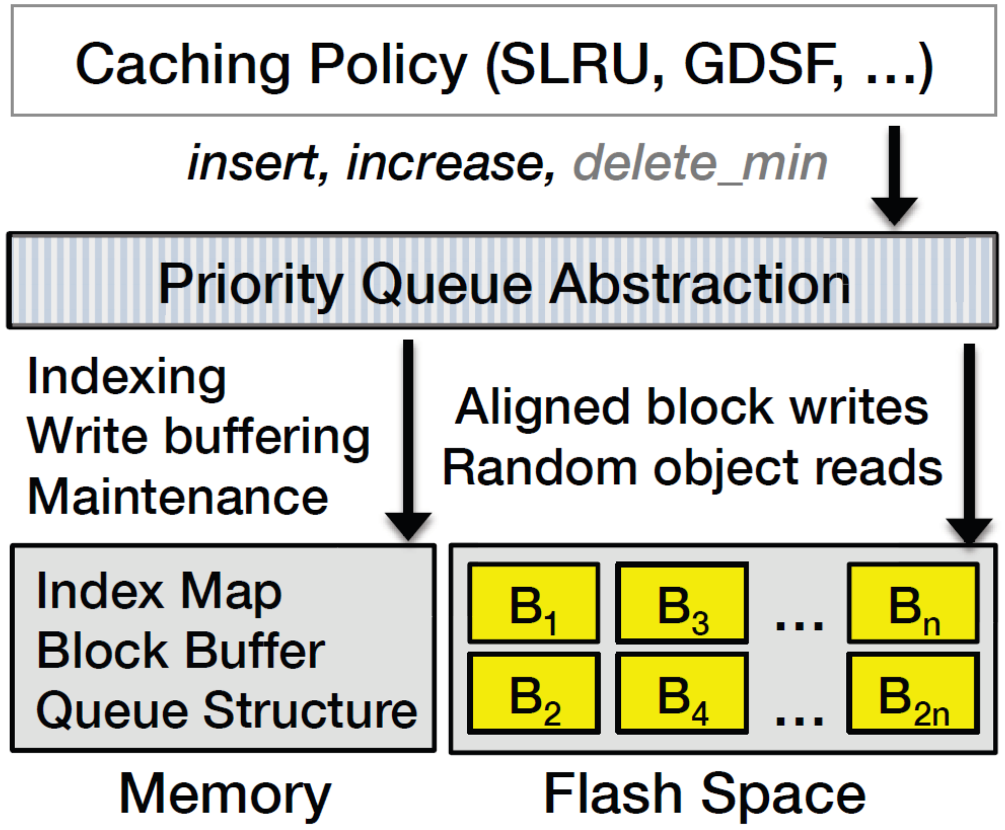
- **Edge caches** close to users
  - Reducing *traffic to datacenters*:
    caches should maximize **byte-wise** hit ratio
- **Origin cache** in datacenters
  - Reducing *back-end I/Os*:
    caches should maximize **object-wise** hit ratio
- Design a system to support both?



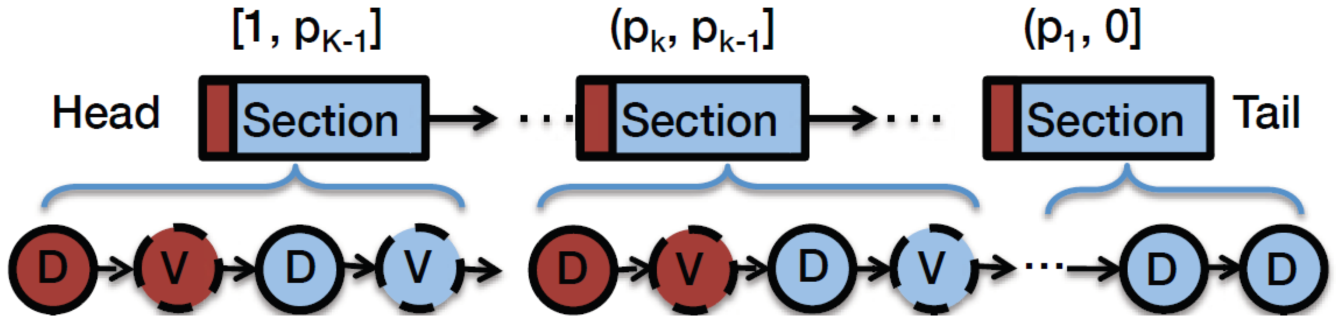
Restricted Insertion Priority Queue (RIPQ)

- General framework to support *many* advanced caching techniques
  - *Segmented-LRU (SLRU)*: better *byte-wise* hit ratio (edge caches)
  - *Greedy-Dual-Size-Frequency (GDSF)*: better *object-wise* hit ratio (origin cache)
- **Priority queue** abstraction – but only an **approximation**!
  - insert(x, p): insert object x with priority p
  - $\text{increase}(x, p)$: increase priority of object x to p
  - $\text{delete-min}(x, p)$: delete object with lowest priority

System Architecture

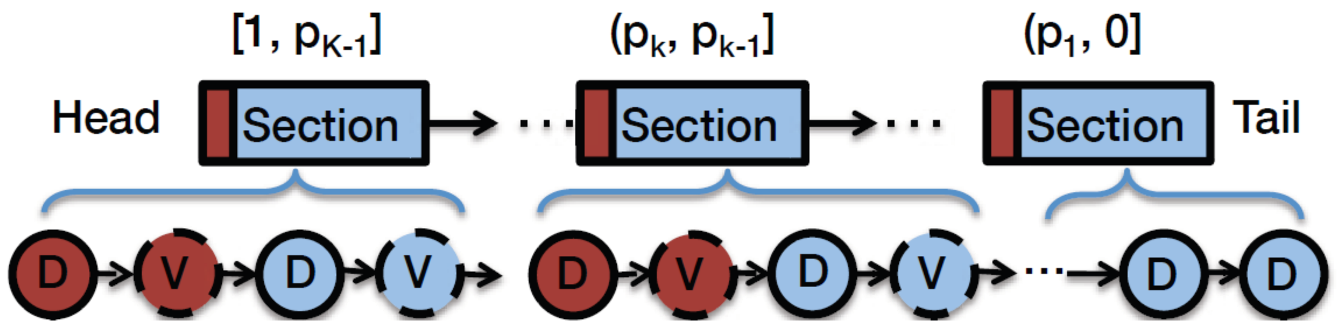


Queue Sections



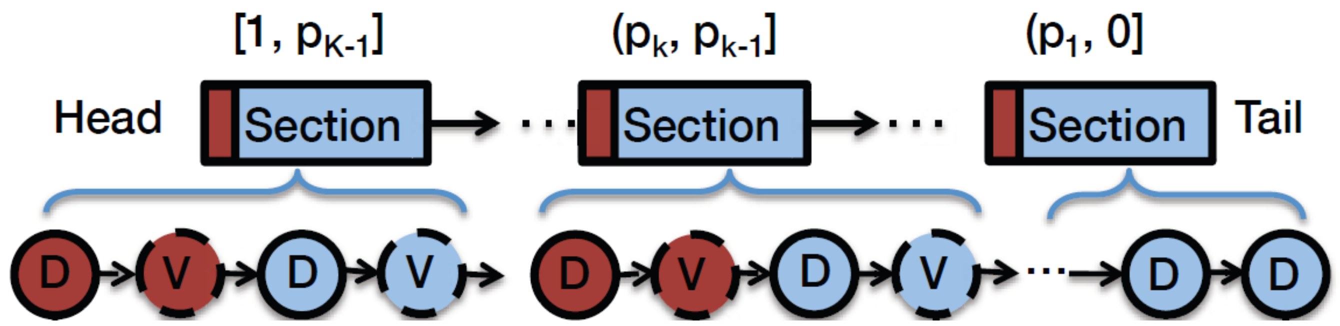
- Divide *priority space* $[0,1]$ into K **sections**, ex. $[1,0.7], (0.7,0.3], (0.3,0]$
  - Sections composed of **blocks** (flash data unit)
- **Insert** new objects at the *head* of a section (based on priority)
  - All objects in *same or lower section* are **implicitly demoted** in the queue
  - Restricted to K insertion points: *accuracy vs. space* tradeoff

Device Blocks



- Each section has *one* **active device block** and *many* **sealed device blocks**
  - *Active block* accepts and **buffers** object insertions in **memory**
  - When active block is full: **flush** contents to **flash**, turn into a *sealed block*
- **Index map** associates objects' keys with location (flash/RAM, block ID, …)

Virtual Blocks

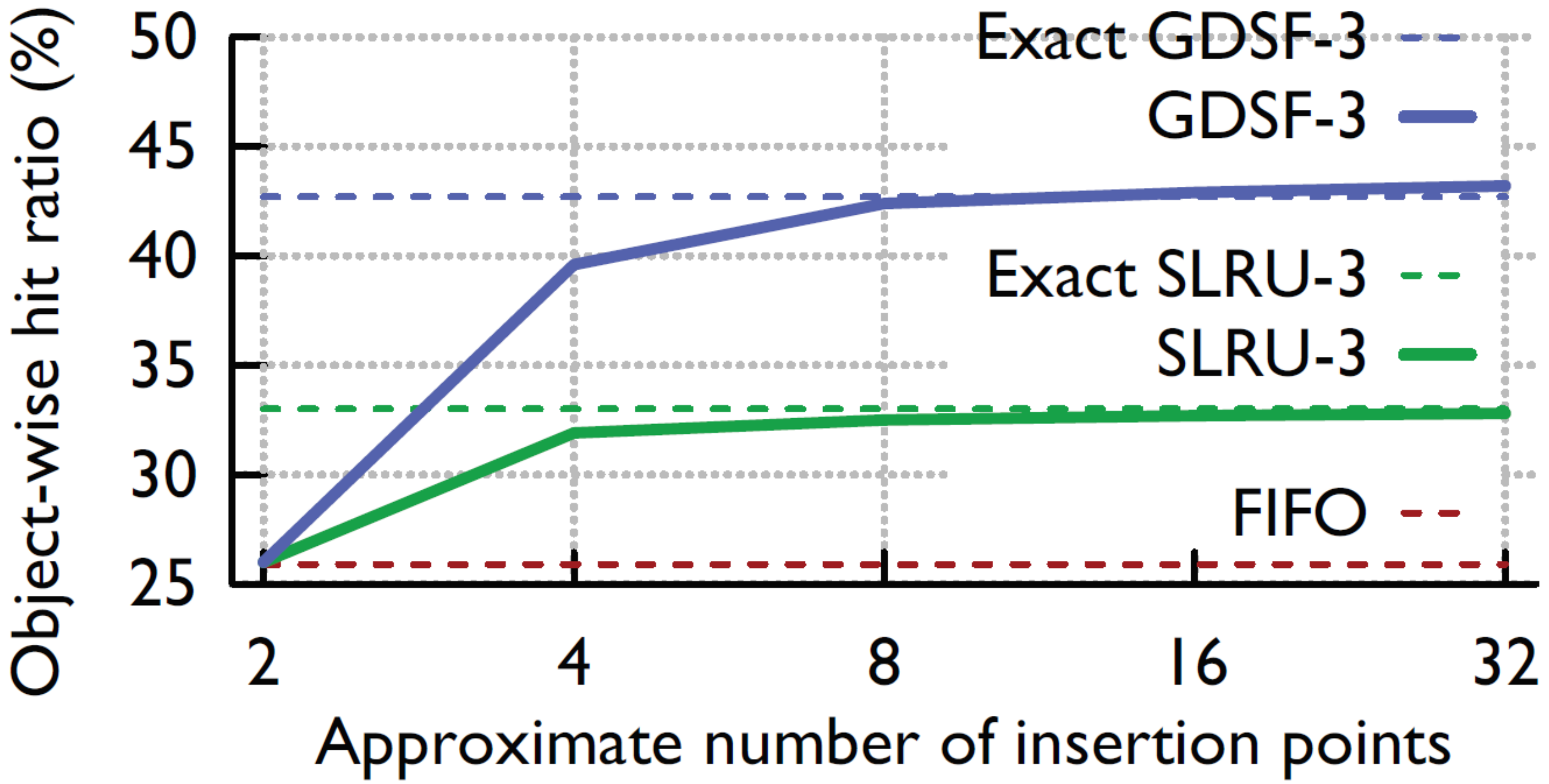


- To handle **priority increases**, use **lazy updates** instead of *duplicating* an object
- Each section also has *one* **active virtual block** and *many* **sealed virtual blocks**
  - Updates only add a *pointer* in the virtual block, and set a *virtual block ID* in the *index map*
  - Upon *eviction* of a *device block*, objects with a *virtual block ID* are *re-inserted* into the corresponding *active device block*

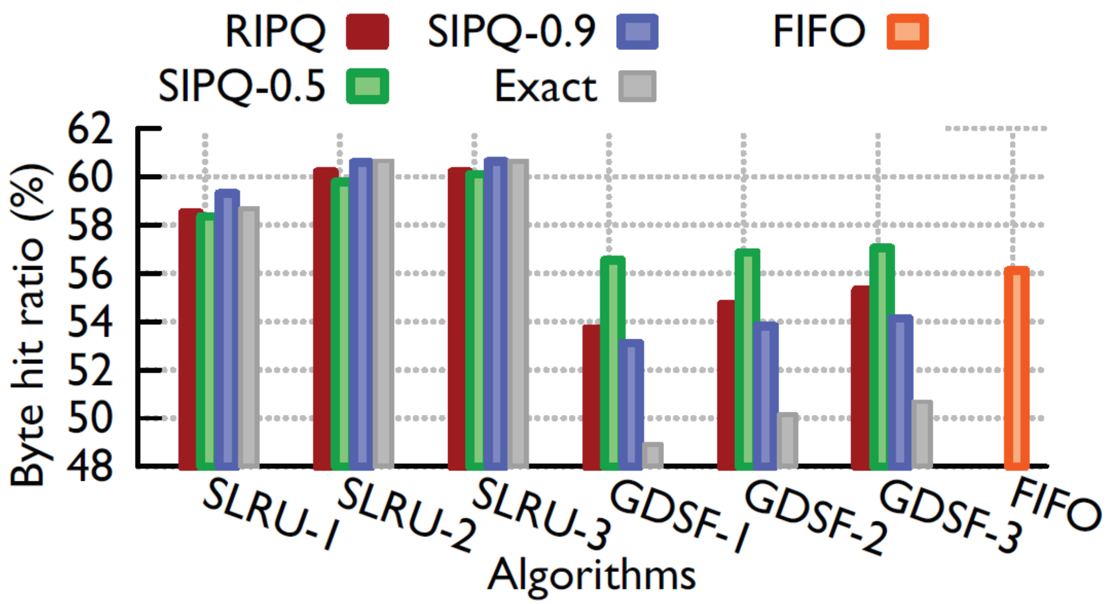
Single Insertion Priority Queue (SIPQ)

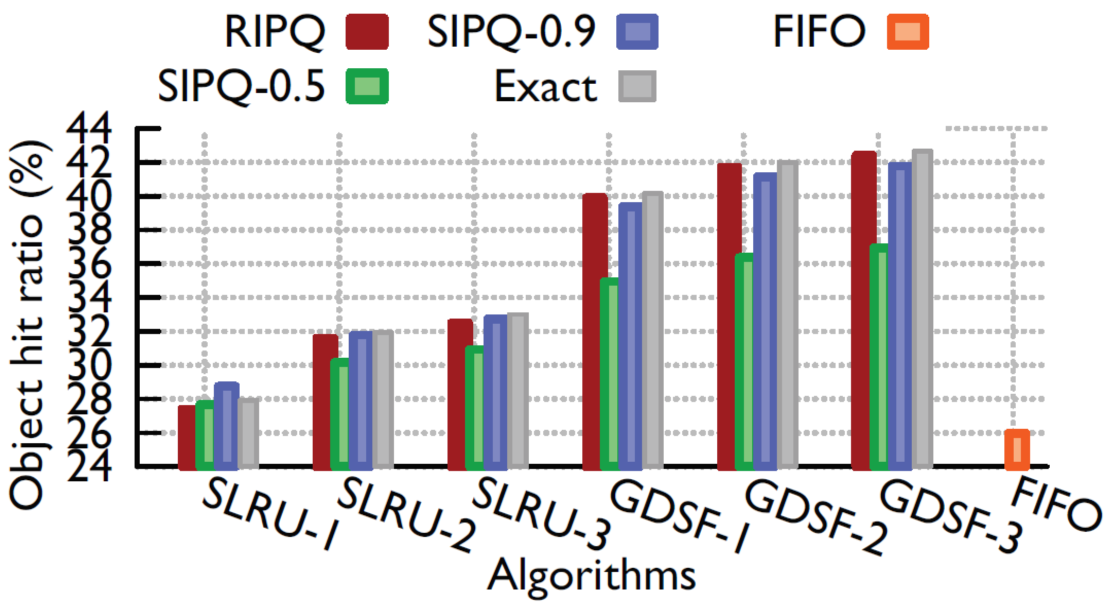
- Simplifies RIPQ for use in memory-constrained settings
- Good results for simpler algorithms (ex. LRU)

Evaluation: Fidelity



Evaluation: Hit Ratios

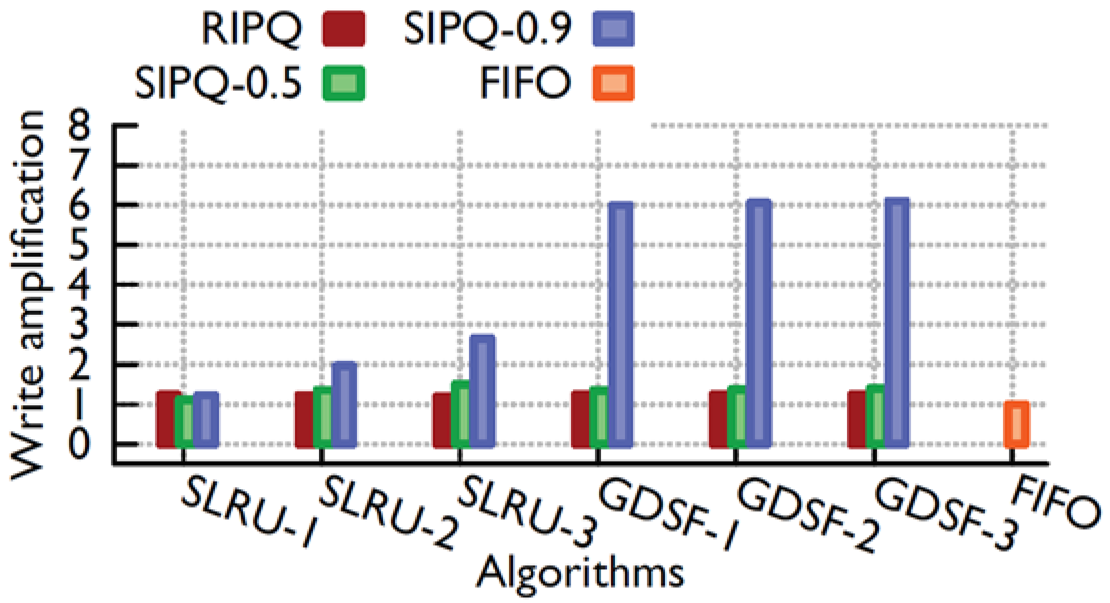




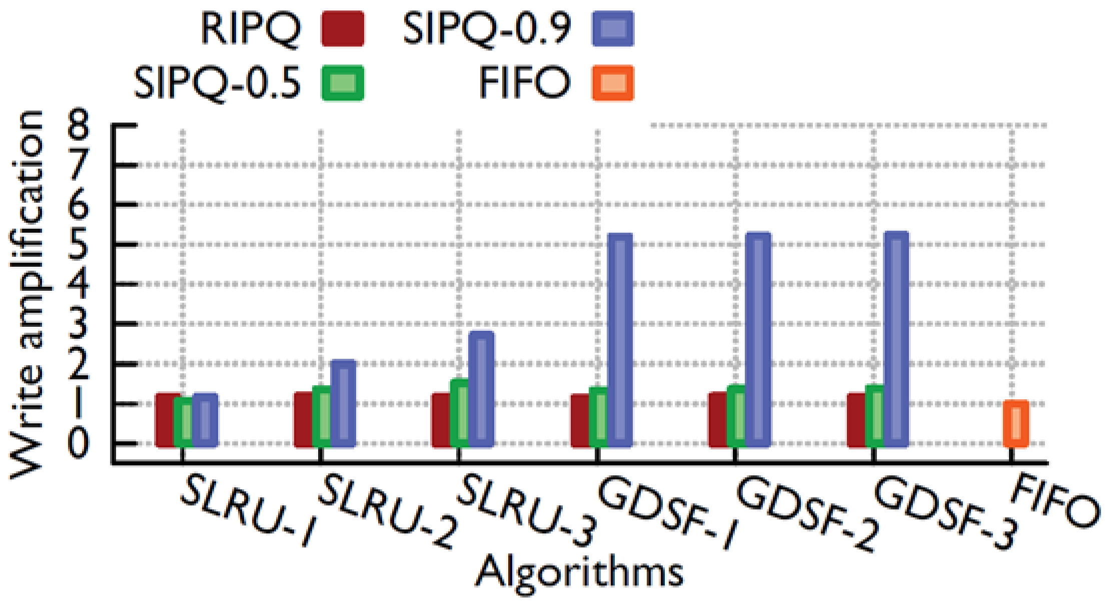
Byte-wise hit ratio (Edge)

Object-wise hit ratio (Origin)

Evaluation: Write Amplification



Write amplification (Edge)



Write amplification (Origin)