RIPQ: Advanced Photo Caching on Flash for Facebook
Linpeng Tang (Princeton), Qi Huang (Cornell & Facebook), Wyatt Lloyd (USC & Facebook), Sanjeev Kumar (Facebook), Kai Li (Princeton)

Motivation
- Billions of photos shared daily on FB
- 2-layer caching network
  - Flash used for both layers
- Advanced algorithms help
  - GDSF-3: 23% fewer backend IO
  - SLRU-3: 10% less backbone traffic
- No known way to implement them efficiently on Flash
- Incurs harmful random writes
- FIFO was still used in production

Advanced Caching Policy
(SLRU, GDSF, ...)

Restricted Insertion Priority Queue
- First framework for advanced caching on Flash
- Efficient implementation of an approximate priority queue on Flash
- Advanced caching algorithms run approximately & efficiently on top
- Generates flash-friendly workloads
- Delivers high caching performance as exact algorithms.

Relative Priority Queue
On a hit: increase(x, p')
On a miss: insert(x, p)

Approximate Insertion
- Assign priorities in 1 .. 0 to relative locations in the queue
- Cache miss: insert(x, p)
- Cache hit: increase(x, p')
- Cache eviction: delete-min()
- Provides an abstraction layer for most advanced algorithms
  - LRU, SLRU, LFU, GDSF, ...

High Approximation Fidelity
- ≤0.5% difference in hit ratio compared to all exact algorithms

Small # of Insertion Points
- All evaluations run on FB workloads
- Sensitivity analysis of # of insertion points on hit ratio
  - Converges at 8 insertion points

High Throughput
- High throughput comparable to FIFO
  - ≤10% difference

Lazy Update
On block eviction:
- copy data to active block
- Virtual blocks to track updated priorities
- Avoid data duplication on priority increase
- Actual update upon eviction
- Always one copy of data

Buffer writes into large blocks for efficiency on Flash
- 256MiB, 90% utilization according to experiments

Restricted Insertion points to a few active blocks
- # of insertion points trades off
  - Insertion accuracy
  - RAM buffer usage

High Throughput comparable to FIFO
- ≤10% difference
- Additional overhead comes from carrying out priority increase

Restricted Insertion Priority Queue