Hot Data Identification for Flash-based Storage Systems Using Multiple Bloom Filters

Dongchul Park and David H.C. Du

Computer Science and Engineering

University of Minnesota, Twin Cities

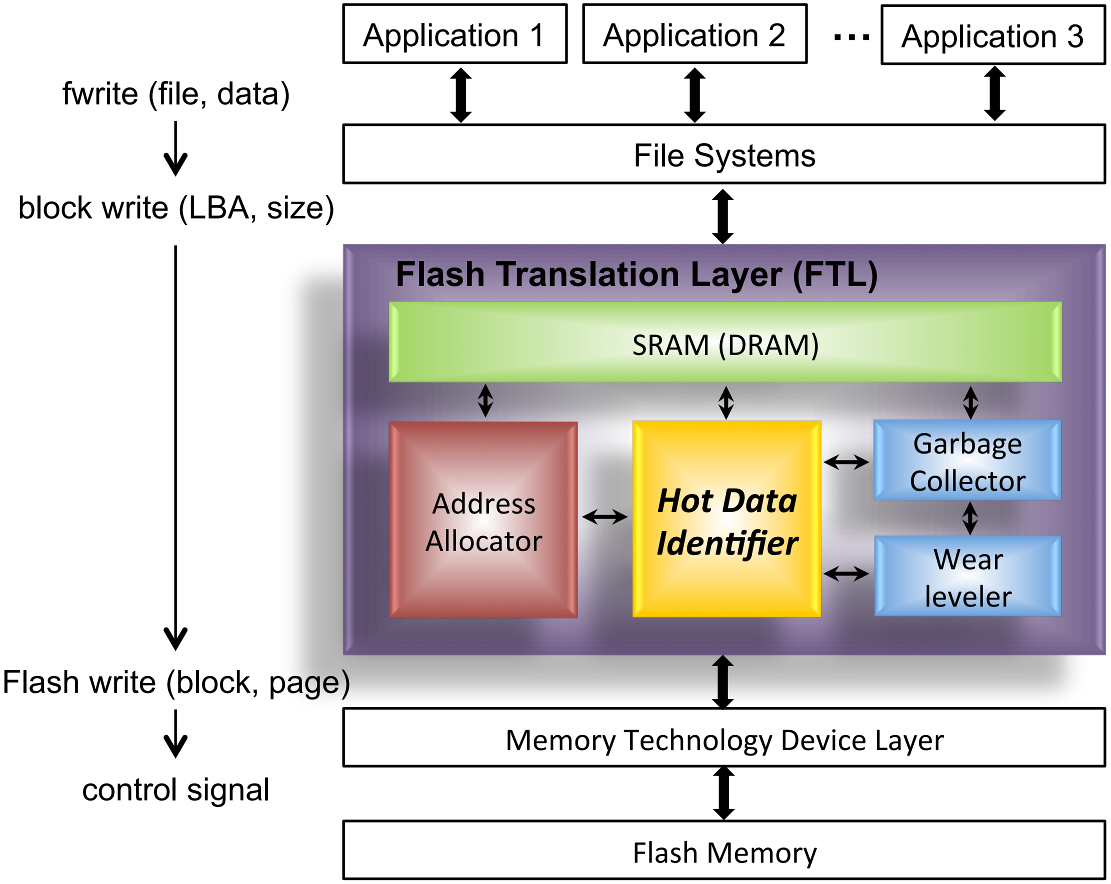
UNIVERSITY OF MINNESOTA

Motivations

- NAND Flash-based Storages
  - Good performance
  - Price is getting cheaper and cheaper
  - Widely adopted to a variety of fields
    - Enterprise servers as well as personal computers

- Applications (flash memory areas)
  - Garbage collection and wear leveling
  - Flash as a cache
  - Address mapping scheme
  - Hybrid SSDs
  - Buffer replacement algorithm
  - Sensor networks, etc...

Architecture of Flash-based Storage



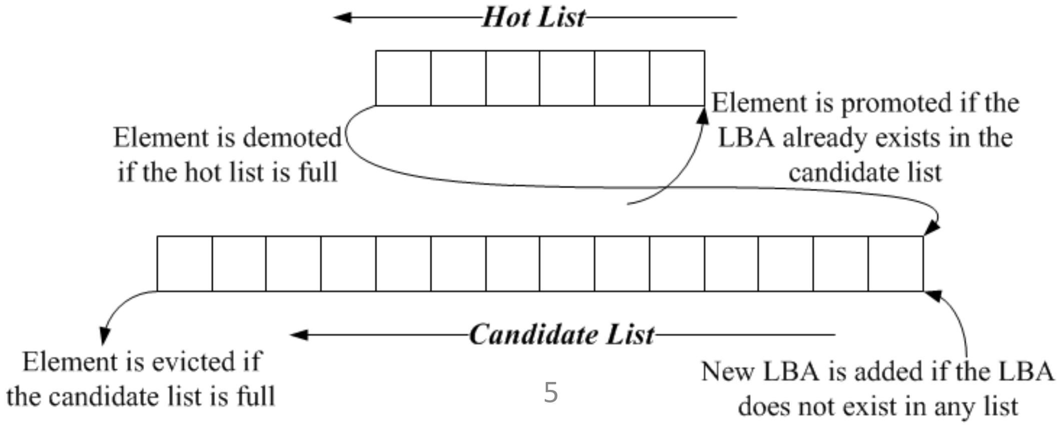
Motivations

- Flash memory
  - Hot data identification has a critical impact on
    - The performance (due to GC)
    - The lifespan (due to WL)
  - Least investigated issue

- Factors for hot data identification
  - **Frequency** and **recency**
  - Not consider recency so much

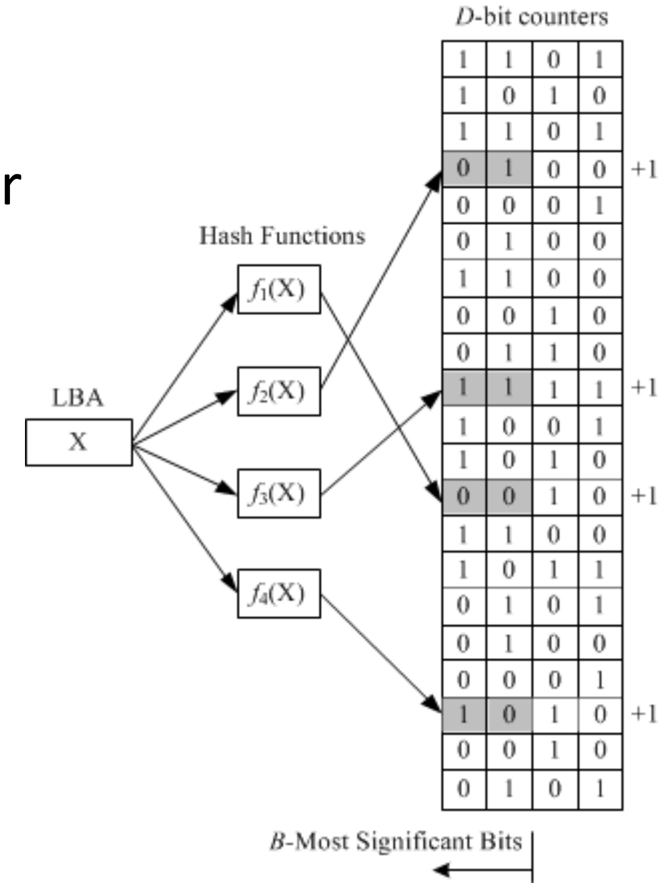
Existing work

- Two-level LRU
  - Maintains a hot list and candidate list
    - Operates under LRU algorithm
    - Save memory space
  - Performance is sensitive to the sizes of both lists
  - High computational overhead


Hot List
Element is demoted if the hot list is full
Element is promoted if the LBA already exists in the candidate list
Candidate List
Element is evicted if the candidate list is full
New LBA is added if the LBA does not exist in any list

Existing work

- ❑ Multi-hash function scheme
  - Adopts multiple hash functions
    - One bloom filter with D-bit counter
    - Decay after a specific period
    - Save memory space and computational overheads
  - Baseline (ideal) algorithm
    - DAM (Direct Address Method)
  - *Cannot capture recency*
    - *Exponential batch decay*



1	1	0	1	
1	0	1	0	
1	1	0	1	
0	1	0	0	+1
0	0	0	1	
0	1	0	0	
1	1	0	0	
0	0	1	0	
0	1	1	0	
1	1	1	1	+1
1	0	0	1	
1	0	1	0	
0	0	1	0	+1
1	1	0	0	
1	0	1	1	
0	1	0	1	
0	1	0	0	
0	0	0	1	
1	0	1	0	+1
0	0	1	0	
0	1	0	1	

Efficient hot data identification

- Requirements
  - Effective recency capture
    - As well as frequency capture
  - Small memory consumption
  - Low runtime overheads

- **Goal**: design an efficient hot data identification scheme

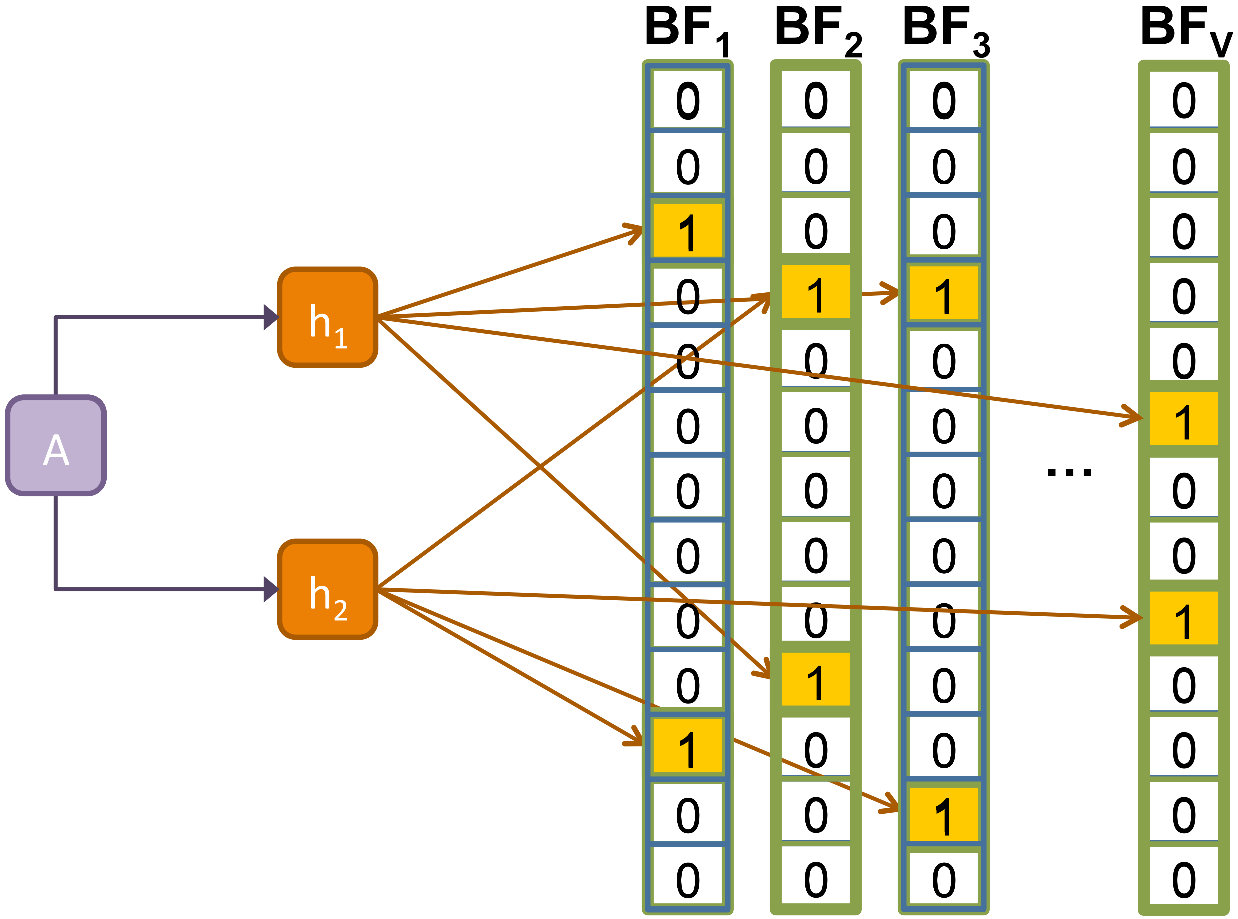
→ ***Multiple bloom filter-based hot data identification scheme***

→ ***WDAC (Window-based Direct Address Counting)***

Multiple BF-based scheme

- ❑ Overview
  - Multiple bloom filters
    - To capture finer-grained recency
    - To reduce memory space and overheads
  - Multiple hash functions
    - To reduce false identification
  - Frequency
    - Does not maintain access counters
  - Recency
    - Different recency coverage

Basic operations



BF_1	BF_2	BF_3	BF_V
0	0	0	0
0	0	0	0
1	0	0	0
0	1	1	0
0	0	0	0
0	0	0	1
0	0	0	0
0	0	0	0
0	0	0	1
0	1	0	0
1	0	0	0
0	0	1	0
0	0	0	0

Capturing Frequency

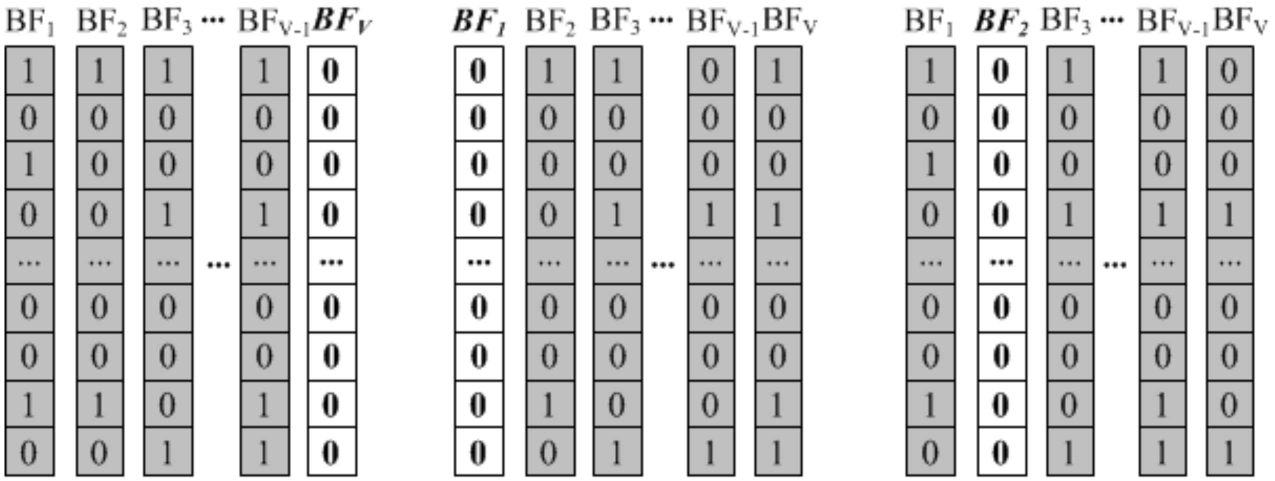
- No access counters
  - Needs a different mechanism
- For frequency capturing
  - Chooses one of BFs in a round-robin manner
  - If the chosen BF has already recorded the LBA
    - Records to another BF available.
  - Shortcut decision
    - If all BFs store the LBA information
      - Simply define the data as hot

→ ***The Number of BF can provide frequency information***

Capturing Recency

- ❑ After a decay period (T)
  - ❖ Choose one of *V*-BFs in a round-robin manner
  - ❖ Erase all information (i.e., reset all bits to 0)

➔ ***Each BF retains a different recency coverage.***



BF$_1$	BF$_2$	BF$_3$	⋯	BF$_{V-1}$	***BF$_V$***
1	1	1		1	**0**
0	0	0		0	**0**
1	0	0		0	**0**
0	0	1		1	**0**
⋯	⋯	⋯	⋯	⋯	**⋯**
0	0	0		0	**0**
0	0	0		0	**0**
1	1	0		1	**0**
0	0	1		1	**0**

(a) First *T* Interval

BF$_1$	BF$_2$	BF$_3$	⋯	BF$_{V-1}$	BF$_V$
0	1	1		0	1
0	0	0		0	0
0	0	0		0	0
0	0	1		1	1
⋯	⋯	⋯	⋯	⋯	⋯
0	0	0		0	0
0	0	0		0	0
0	1	0		0	1
0	0	1		1	1

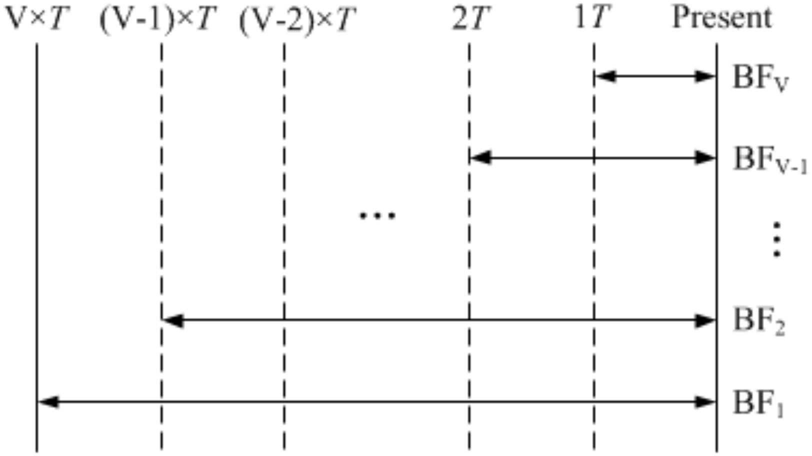
(b) Second *T* Interval

BF$_1$	***BF$_2$***	BF$_3$	⋯	BF$_{V-1}$	BF$_V$
1	**0**	1		1	0
0	**0**	0		0	0
1	**0**	0		0	0
0	**0**	1		1	1
⋯	**⋯**	⋯	⋯	⋯	⋯
0	**0**	0		0	0
0	**0**	0		0	0
1	**0**	0		1	0
0	**0**	1		1	1

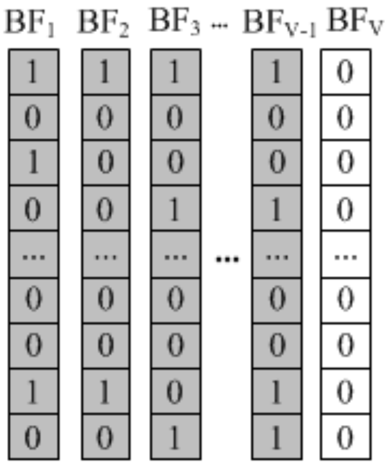
(c) Third *T* Interval

Recency coverage

- ❑ For finer-grained recency
  - ❖ Each BF covers a different recency coverage
    - The reset BF (BF_v): Shortest (latest) coverage
    - The next BF (BF_1): Longest (oldest) coverage
  - ❖ Each BF has a different recency value



(a) Recency Coverage



BF_1	BF_2	BF_3	...	BF_{V-1}	BF_V
1	1	1		1	0
0	0	0		0	0
1	0	0		0	0
0	0	1		1	0
...	...	...	...	...	...
0	0	0		0	0
0	0	0		0	0
1	1	0		1	0
0	0	1		1	0

(b) Bloom Filter Status

Discussion: Baseline Algorithm

- DAM (Direct Address Method)
  - An existing baseline algorithm
  - Assuming unlimited memory space and all LBAs maintain their own counters.
  - Retains the same problem as Multi-hash scheme
    - Cannot capture recency properly.

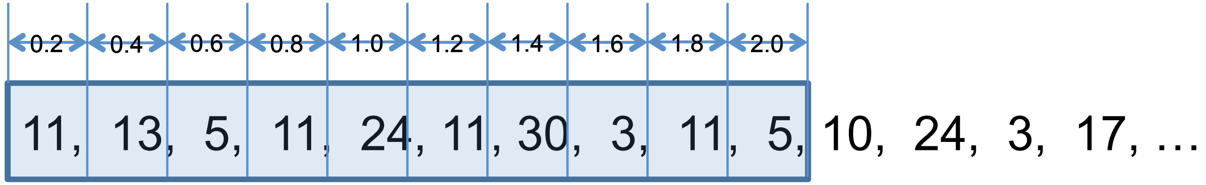
➔ ***Window-based Direct Access Counting (WDAC)***

WDAC

- Overview
  - Adopts a sliding window concept
  - Within the window, *all* elements have a *different* recency value.
    - Head: Highest recency value
    - Tail: Lowest recency value
  - Maintains hot data index (HDI) values

➔ ***WDAC can capture very-fine grained recency information***

WDAC Operations



0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0

11, 13, 5, 11, 24, 11, 30, 3, 11, 5, 10, 24, 3, 17, …

LBA	HDI
11	2.0
30	0.8
5	1.4
24	2.2
3	3.0
10	1.6

Performance Evaluation

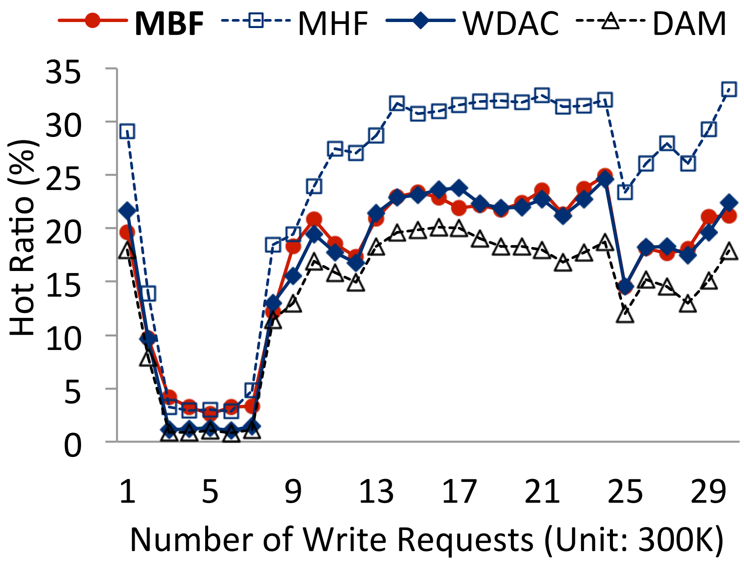
- Evaluation setup
  - Four schemes
    - **Multiple bloom filter scheme (refer to as MBF)**
    - Multiple hash function scheme (refer to as MHF)
    - Direct Address Method (refer to as DAM)
    - Window-based Direct Address Counting (refer to as WDAC)
  - Four realistic workloads
    - Financial1, MSR (*prxy volume 0*), Distilled, and RealSSD

Performance Evaluation

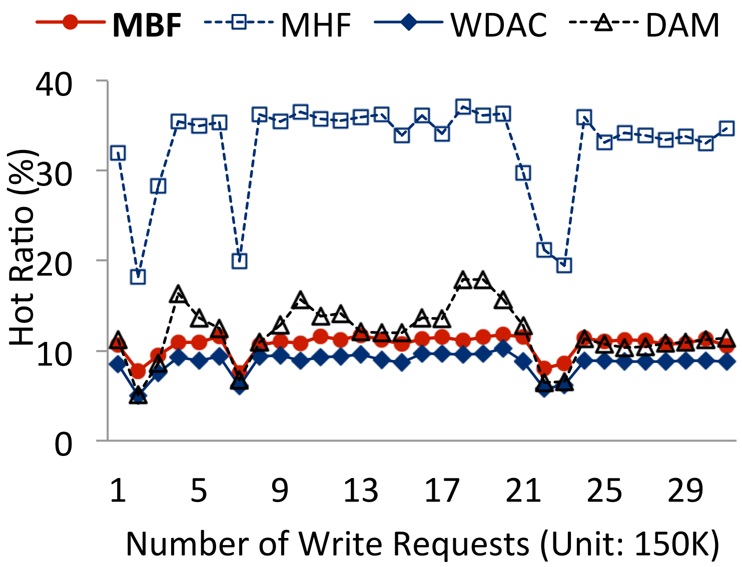
- Performance metrics
  - *Hot ratios*
    - A ratio of hot data to all data
  - *False identification rate*
    - Try to compare each identification result of each scheme whenever a request is issued.
  - *Memory consumption*
  - *Runtime overhead*
    - Measure CPU clock cycles per operation

Hot Ratios

(MBF vs. MHF vs. WDAC vs. DAM)



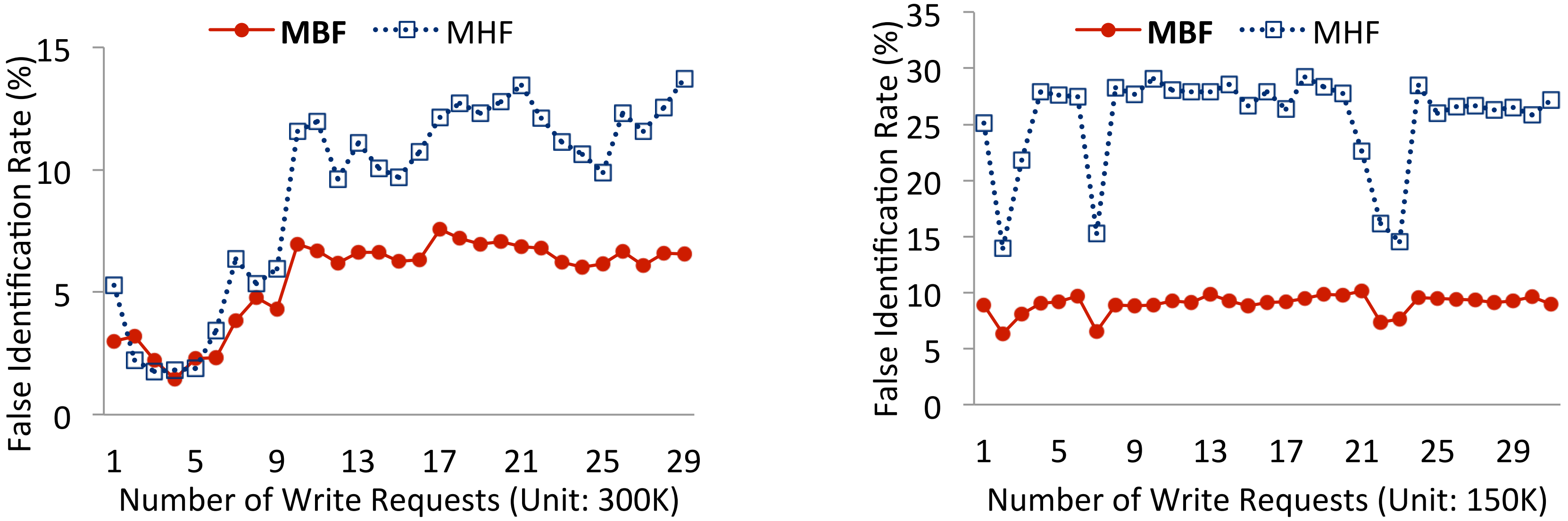
(a) Financial1



(b) MSR

False Identification Rates

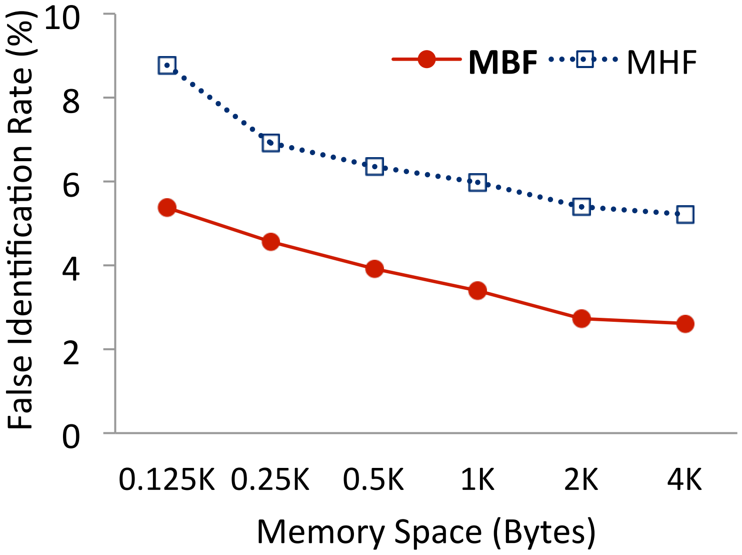
(MBF vs. MHF)

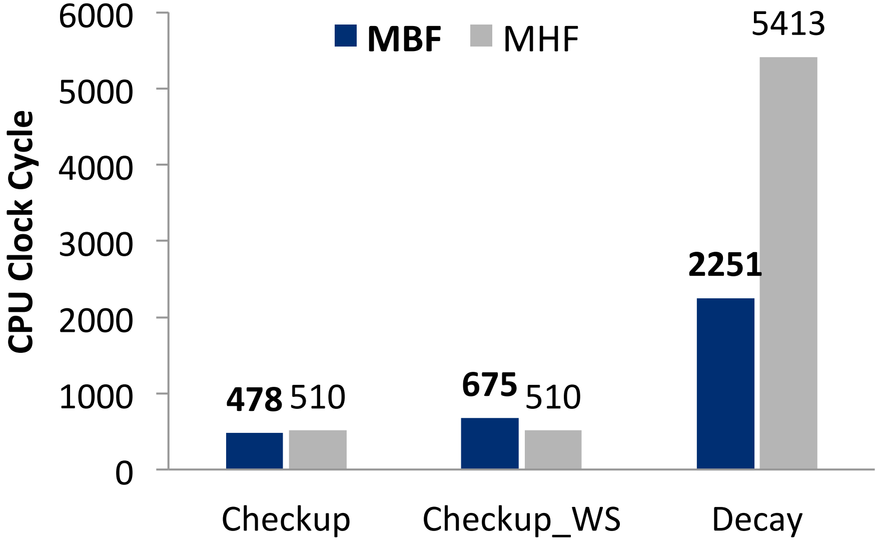


(a) Financial1

(b) MSR

Memory Impact and Computational Overheads





Conclusion

- Multiple BF-based hot data identification
  - Adopts multiple BFs and hash functions
  - Can capture finer-grained recency
  - Shows better performance than multi-hash function scheme
- Window-based Direct Access Counting (WDAC)
  - Adopts a sliding window and maintains hot data index values
  - All elements have a different recency value
  - Can capture very-fine grained recency information

Questions?

- ❑ Thank you!

Dongchul Park

park@cs.umn.edu