Exploring Skyrmion Racetrack Memory for High Performance Full-Nonvolatile FTL

Ya-Hui Yang, Yu-Pei Liang, Cheng-Hsiang Tseng, Shuo-Han Chen

Presented by Ya-Hui Yang,
At NVMSA,
2021.08.18,

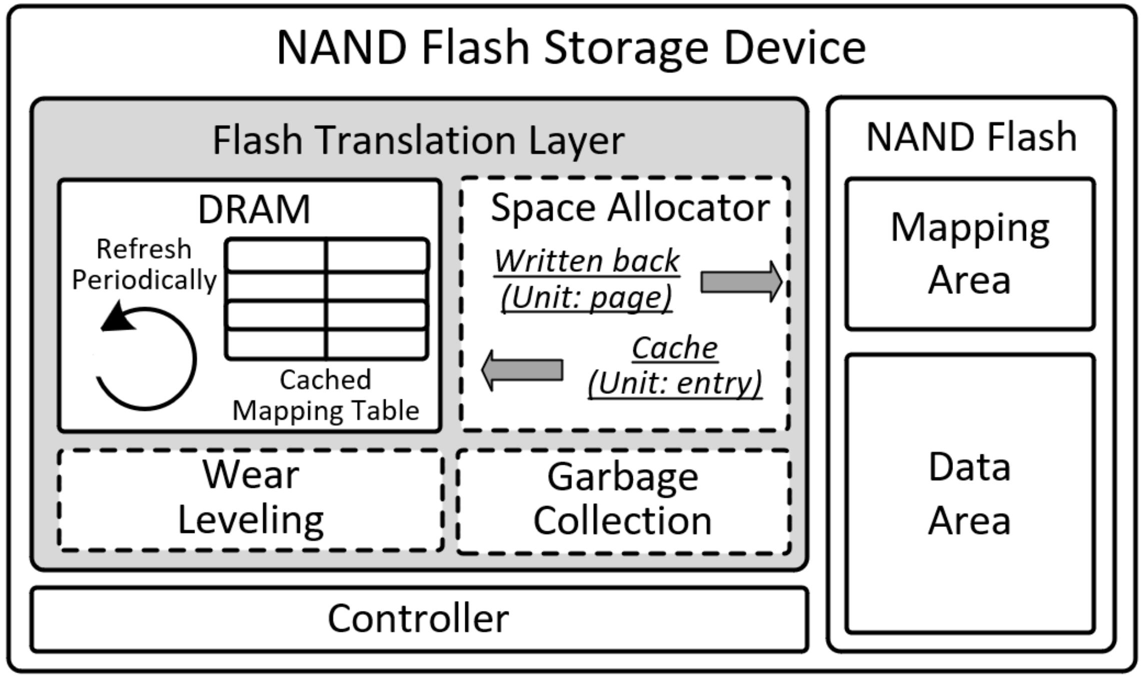
Outline

Overview

- SK-RM has demonstrated great potential as high-density and low-cost NVRAM
  - Support random information update, deletion, and insertion at bit level
  - Fast update / deletion speed
- SK particles are placed on a racetrack with 4 operations
  - Insert, Delete, Detect, Shift
- Proposes a SK-FTL to preserve skyrmions in unused memory space through both horizontal and vertical shifts
- Reduce the accumulated insert and remove latencies of skyrmions by an average of 62.69% and 93.25%, when compared with native page-based FTL on SK-RM with permutation write enabled

Background: Flash Translation Layer (FTL)

- Constraints of NAND flash
  - (1) Erase-before-write (2) Limited program/erase (P/E) cycles (3) Asymmetric access/erase unit
- FTL is used on flash-based storage devices to hide the constraints of NAND flash
  - Maintains a logical address to physical address mapping
  - Mapping entries are loaded & stored on DRAM and NAND Flash
- Components of FTL
  - Space allocator
  - Wear leveling
  - Garbage collection
- Main issues:
  - Movement overhead mapping entries
  - Possible corruption due to volatile DRAM

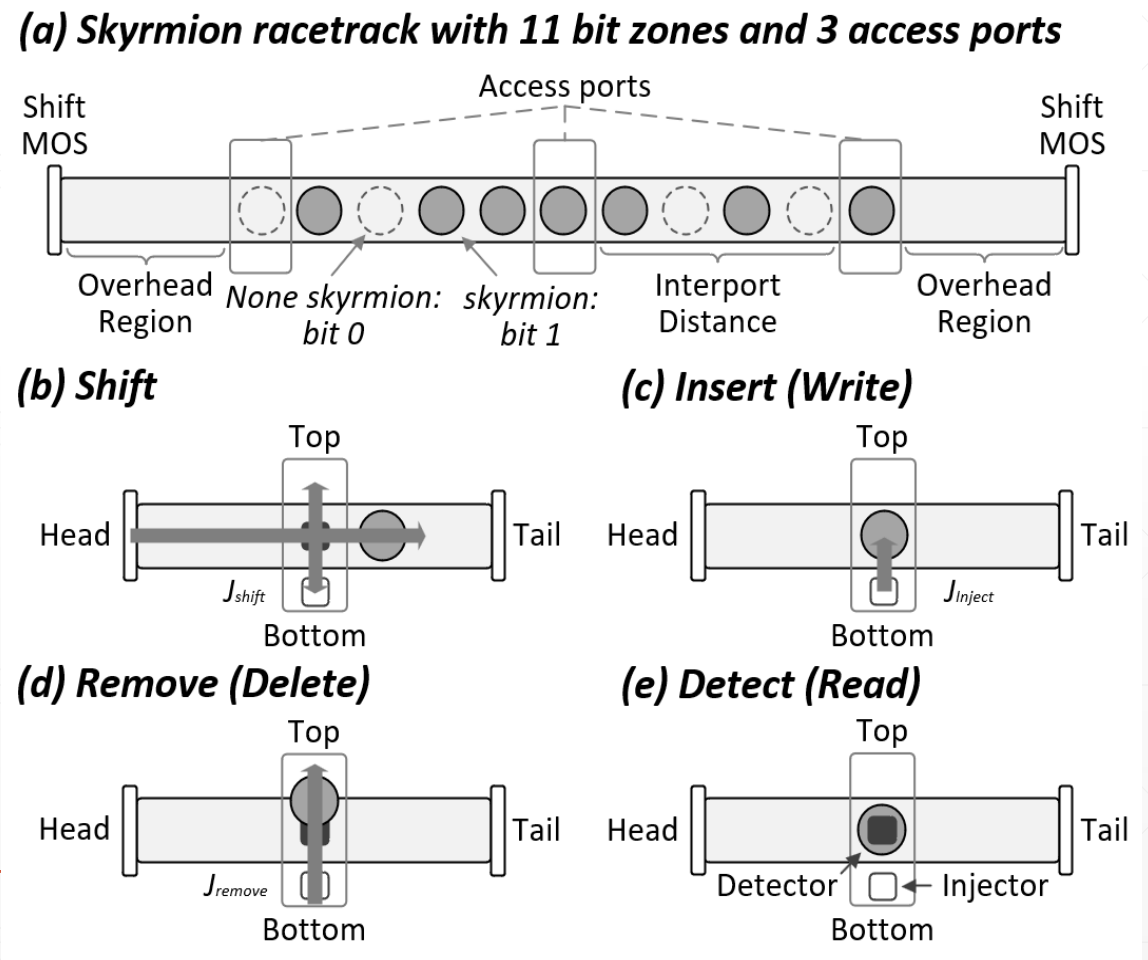


Background: NVRAM-based FTL

- Avoid the load/store of mapping entries between DRAM and NAND flash
- **PCM FTL** – Goal: Lifetime of PCM
  - Storing mapping on PCM to completely replace DRAM
  - enhancing the endurance of the PCM by minimizing the number of bit flips
  - Wear leveling by moving block mapping across PCM
- **NS-FTL / Load-FTL** – Goal: Intra / Inter-entry wear leveling of NVRAM
  - Windows based wear leveling
- **Hybrid FTL** – Goal: Performance
  - Storing mapping on PCM
  - Still include DRAM for caching
- Previous designs haven’t explored SK-RM for high performance FTL
  - Skyrmions can be preserved for future use
  - The vertical shift feature of SK-RM

Background: Skyrmion Racetrack Memory

- Presence of skyrmion: bit 1 & Absence of skyrmion: bit 0
- Access port: contains an injector and a detector
- Overhead regions: for temporarily holding skyrmions
- Skyrmions are placed on a racetrack with 4 operations
  - Shift, Insert, Remove, Detect
- Main challenges:
  - Operations can only be performed at access ports
  - The latency of Insert/delete is longer than other operations



Motivation

- Using FTL on SK-RM
  - Advantage: short read/write latency
  - To be improved: the bit pattern difference between successive FTL entry updates could lead to excessive insert and remove operations
- For example, in Figure (a), the data chunk C is updated twice
- **Conventional page allocation**: pages of each block are <u>allocated sequentially</u> for new data writes
- **Goal**: to preserve and reuse injected skyrmions properly while avoiding additional insert and remove operations

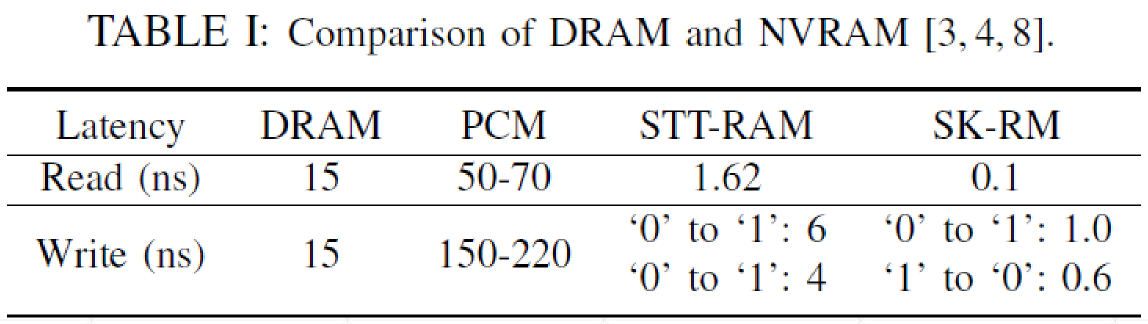
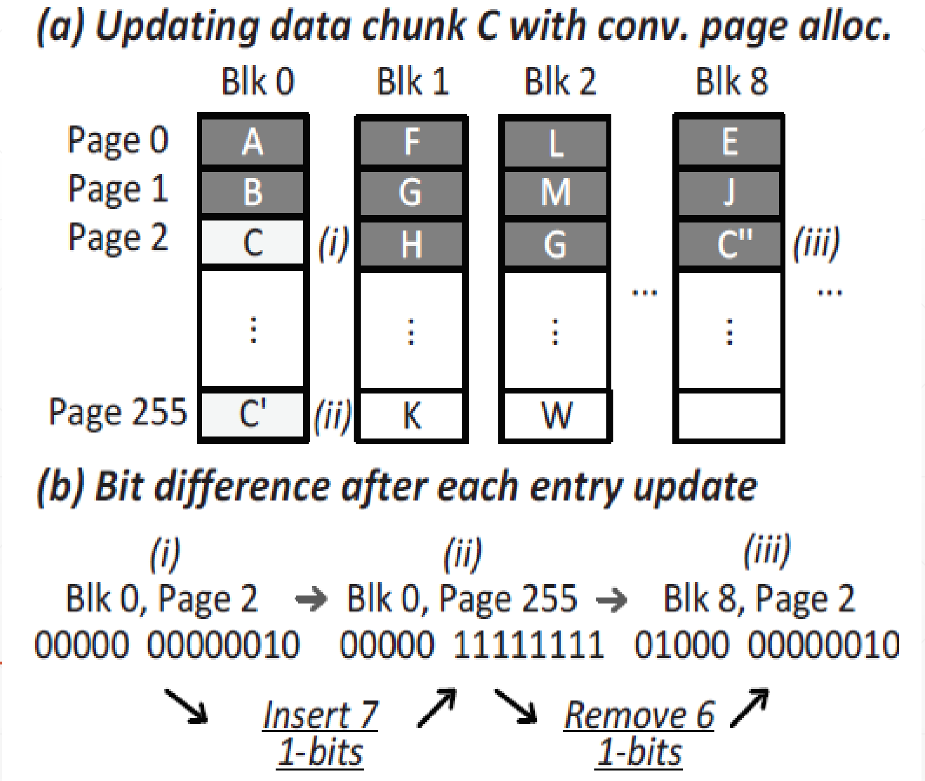


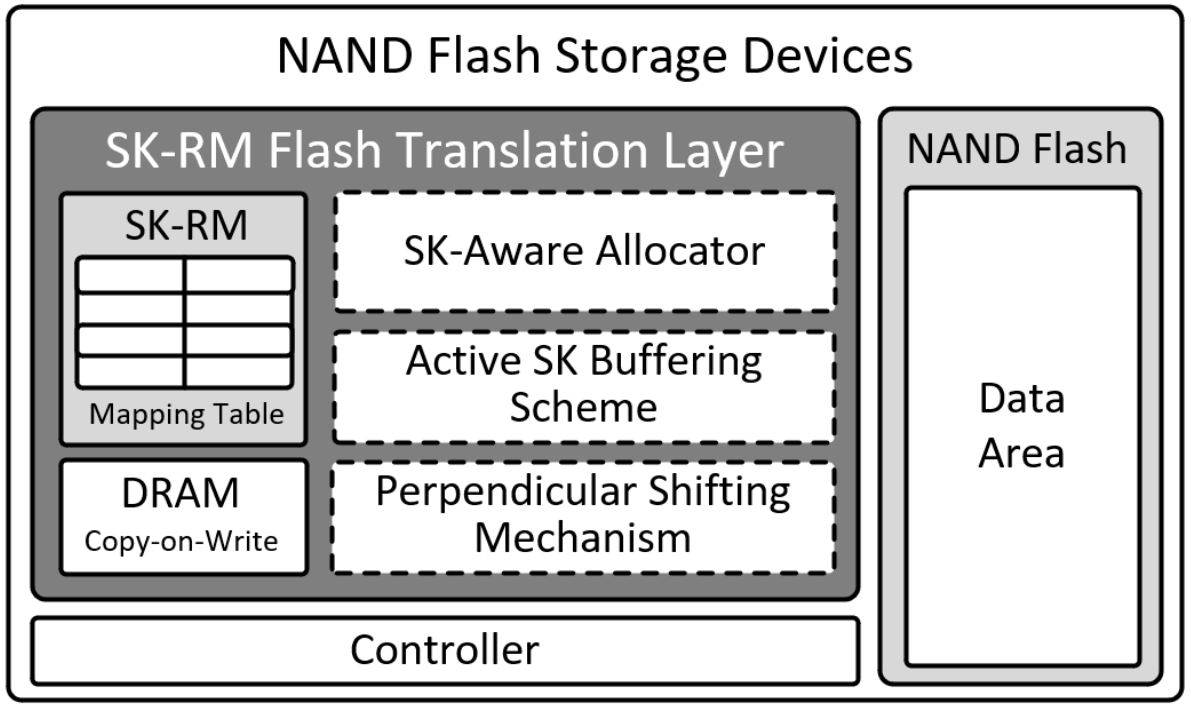
TABLE I: Comparison of DRAM and NVRAM [3, 4, 8].

Latency	DRAM	PCM	STT-RAM	SK-RM
Read (ns)	15	50-70	1.62	0.1
Write (ns)	15	150-220	'0' to '1': 6 '0' to '1': 4	'0' to '1': 1.0 '1' to '0': 0.6



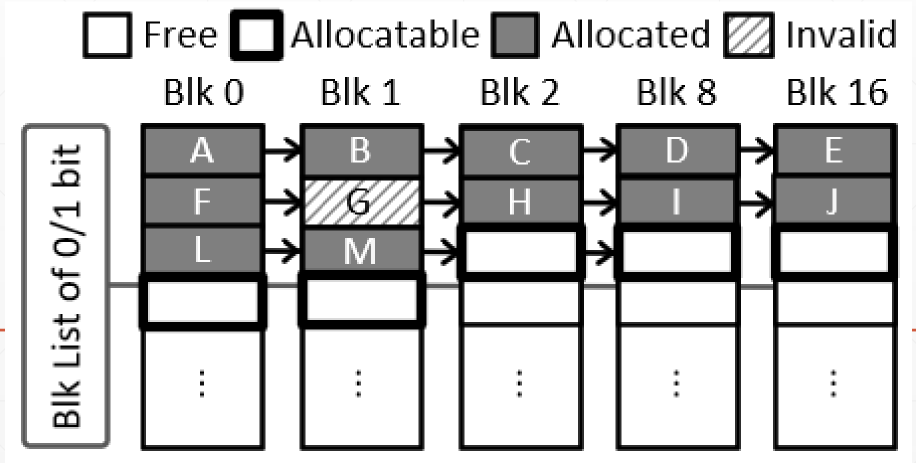
SK-RM-based FTL: Review

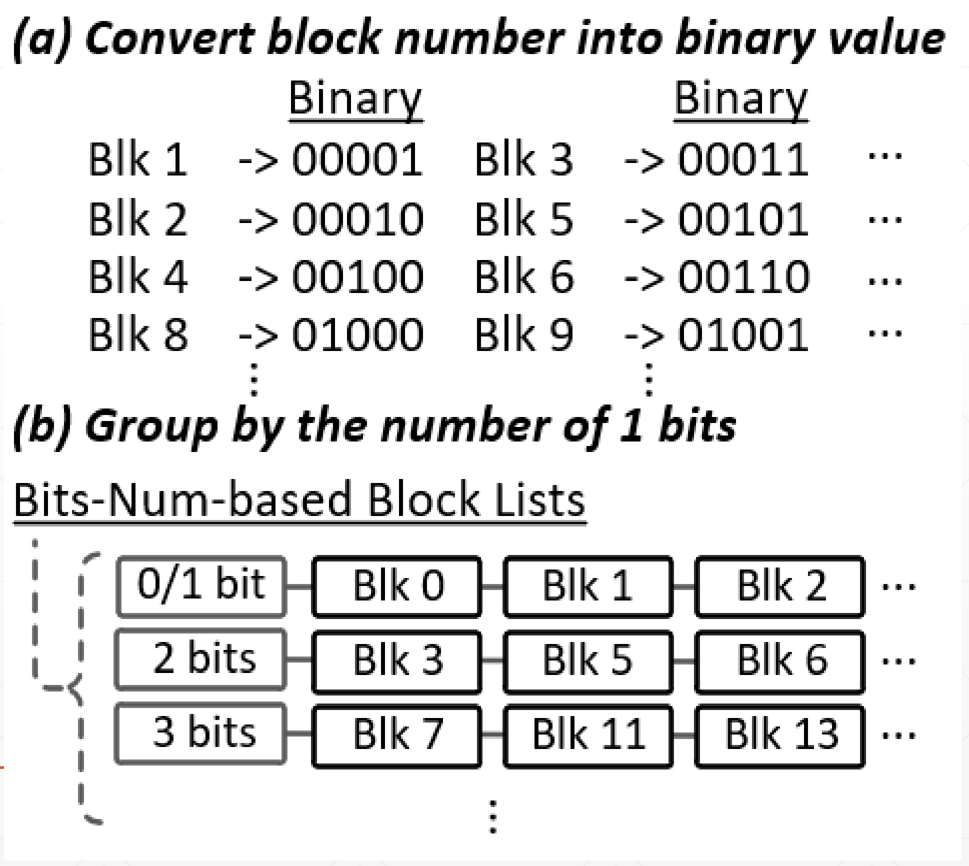
- Explore SK-RM for high performance NVRAM-based FTL
  - Injected SK particles can be retained for future data writes
- Difficulties
  - How to minimize the number of insert & remove operations
  - How to preserve skyrmions for future use
- SK-FTL Design
  - Page-based FTL
  - Each mapping entry fits into the distance
- Components
  - **SK-Aware Space Allocator**
    - Minimize the number of SK insertion
  - **Vertical Shifting Mechanism**
    - Shifting SK between racetracks
  - **Active SK Buffering Scheme**
    - Utilizing free/invalid space for buffering SK


NAND Flash Storage Devices
SK-RM Flash Translation Layer
SK-RM
Mapping Table
SK-Aware Allocator
Active SK Buffering Scheme
DRAM
Copy-on-Write
Perpendicular Shifting Mechanism
NAND Flash
Data Area
Controller

SK-Aware Space Allocator

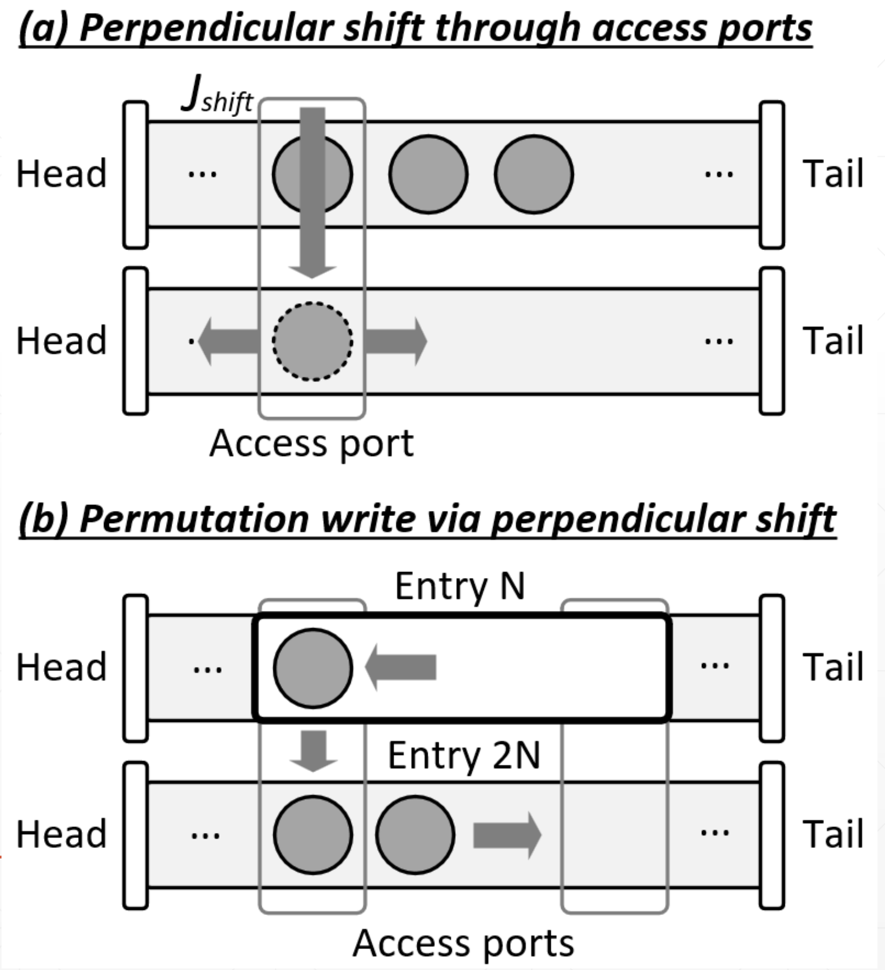
- Updates mapping entries by recomposing skyrimons in overhead region/free space
- To minimize the bit difference, grouping blocks into different lists based on their number of 1 bits in their block addresses
- After considering block addresses, allocate a free page horizontally in the same bits-number-based block list





Vertical Shift Mechanism

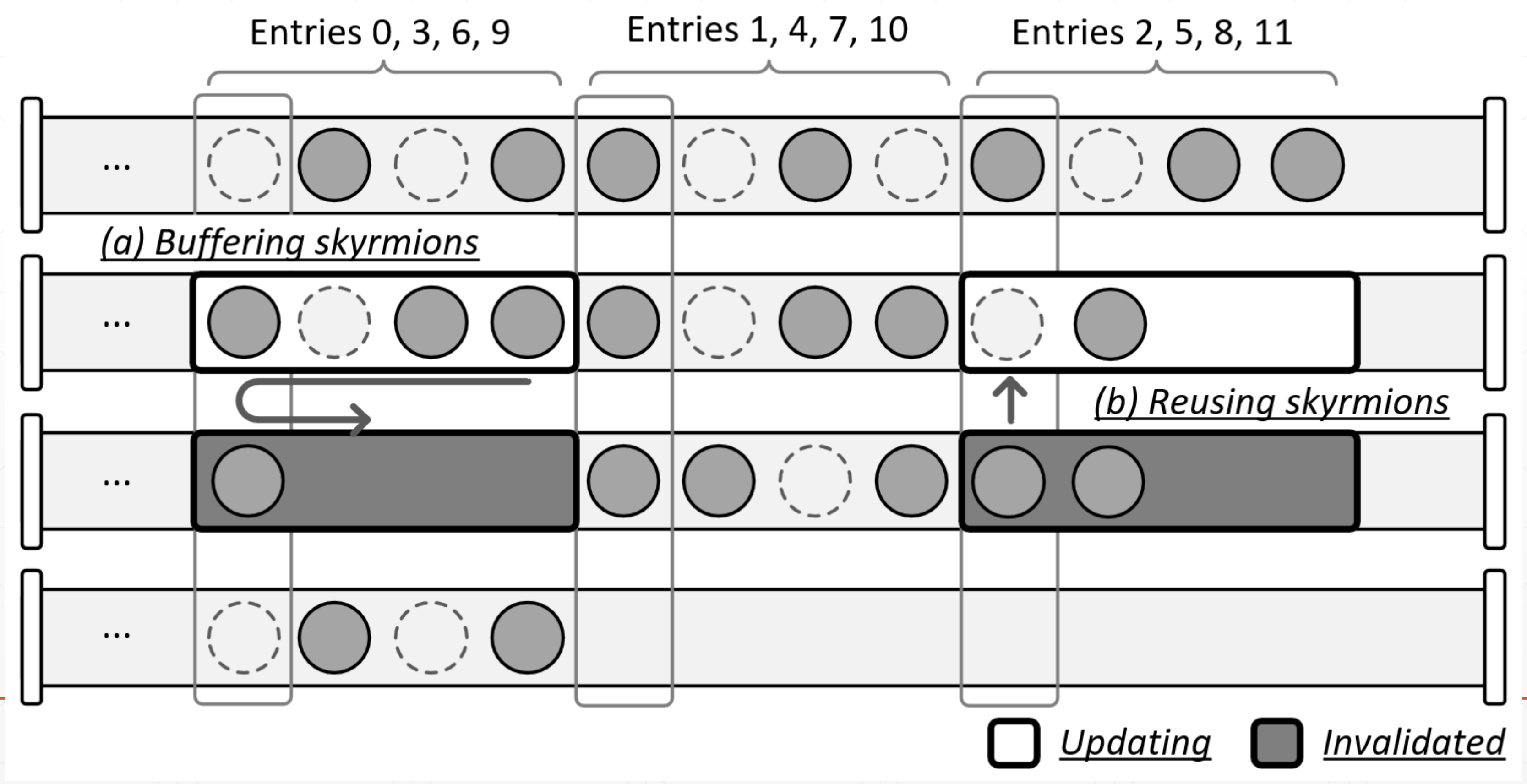
- **Vertical shift**: shifting the skyrmions to adjacent racetracks through the access ports
- Use other free or invalid regions on adjacent racetracks as the buffer region of permutation write strategy
- Since the memory space used as buffer region is free or invalid, excessive skyrmions between mapping entry updates can be preserved
- When skyrmion injections are required during entry updates, those preserved skyrmions in the buffer region can be reused



(a) Perpendicular shift through access ports
J_{shift}
Head
…
…
Tail
Head
…
Tail
Access port
(b) Permutation write via perpendicular shift
Entry N
Head
…
…
Tail
Entry 2N
Head
…
…
Tail
Access ports

Active SK Buffering Scheme

- Repurpose free or invalid mapping entries as buffer region
- Both on-track and inter-track buffering
  - Buffer or reuse through the horizontal or vertical shift operations



Experimental Setup

- Comparisons: FTL, PW FTL, and SK-FTL
  - **FTL**: the FTL on SK-RM with the naive write strategy
    - remove all previous-injected skyrmions and inject new skyrmions based on the updated data pattern
  - **PW-FTL**: FTL on SK-RM with the permutation-write strategy
  - **SK-FTL**: our proposed SKFTL
- Traces: Microsoft Research Cambridge (MSR) & one-month I/O behavior of a personal computer
- The size of the simulated flash is 64 GB with 16KB pages

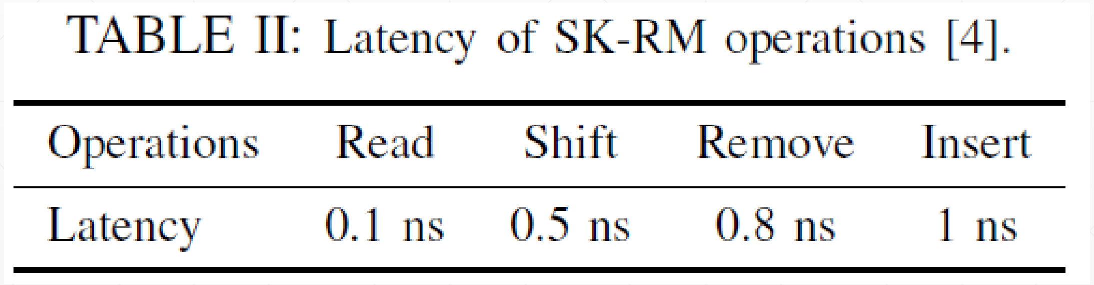


TABLE II: Latency of SK-RM operations [4].

Operations	Read	Shift	Remove	Insert
Latency	0.1 ns	0.5 ns	0.8 ns	1 ns

Experimental Results

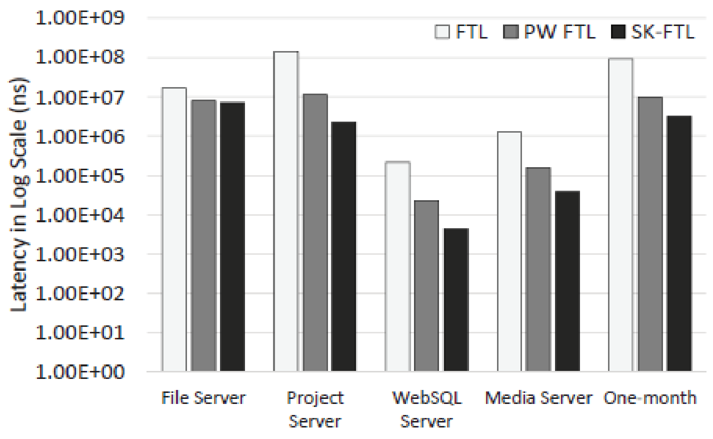


Fig. 9: Inject latency comparison.

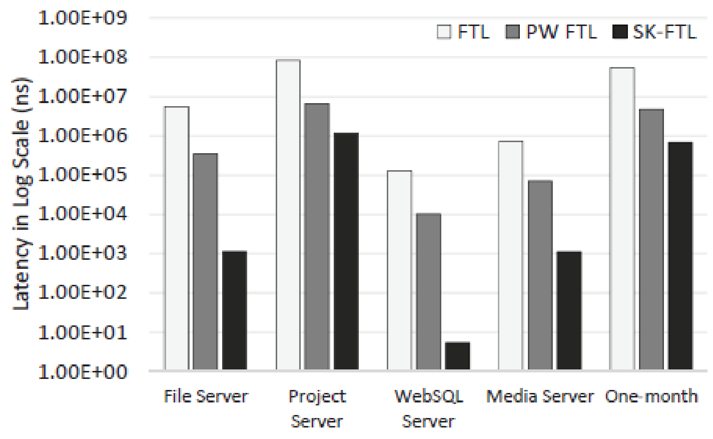


Fig. 10: Remove latency comparison.

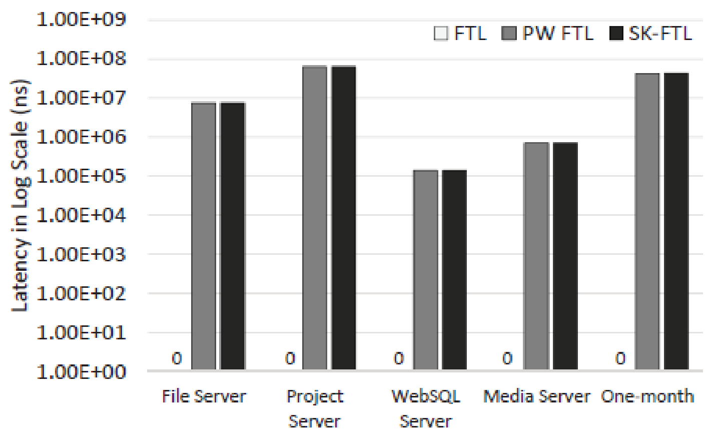


Fig. 11: Detect latency comparison.

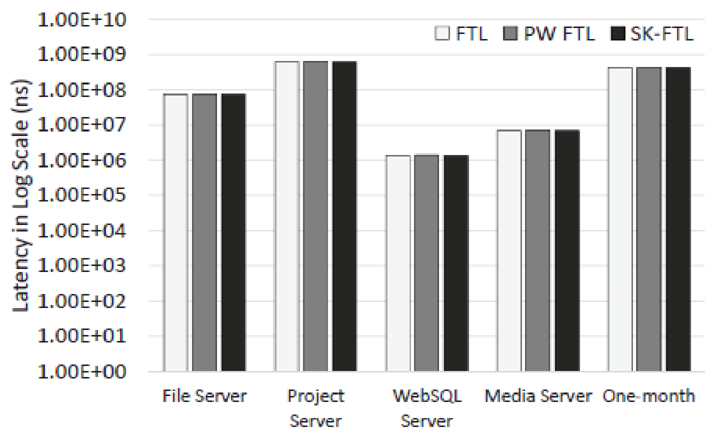


Fig. 12: Shift latency comparison.

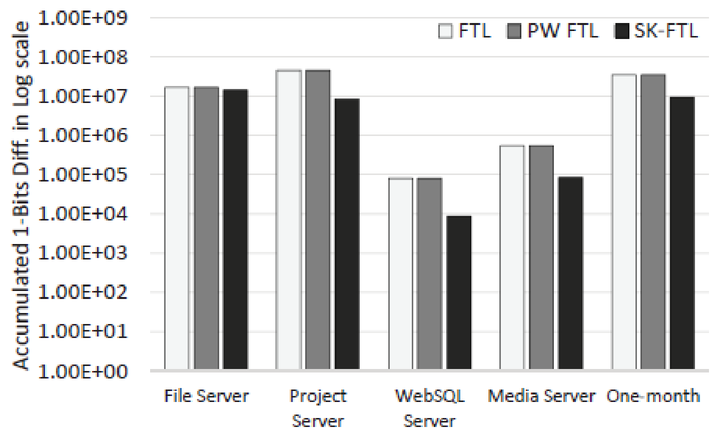


Fig. 13: Accumulated 1-bit difference.

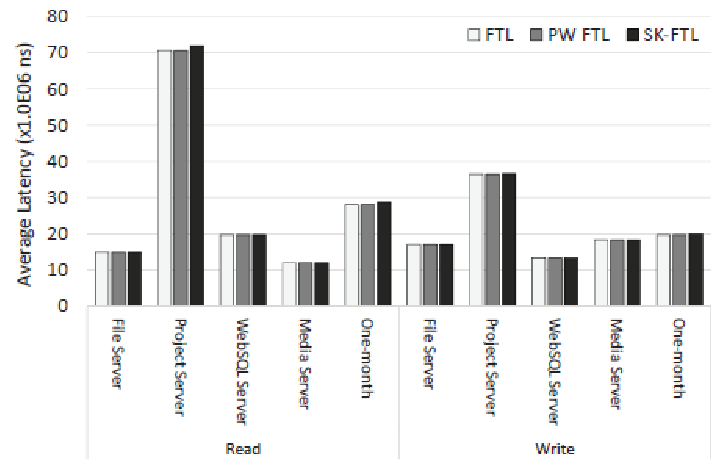


Fig. 14: Read/Write Latency of NAND flash.

Conclusion

- Proposed SK-FTL
  - SK-aware space allocator: minimize the bit-1 difference between mapping updates
  - Vertical shift mechanism
  - The active SK buffering scheme: preserve skyrmions in unused space and recompose skyrmions for future data writes
- Evaluation results: the proposed SK-FTL can reduce the latencies of skyrmion insert and remove operations by 62.69% and 93.25% on average compared with the PW FTL

Thank you for your listening.