WARM

Improving NAND Flash Memory Lifetime with Write-hotness Aware Retention Management

***Yixin Luo**, Yu Cai, Saugata Ghose, Jongmoo Choi*, Onur Mutlu*

*Carnegie Mellon University, *Dankook University*

SAFARI

Carnegie Mellon

DANKOOK UNIVERSITY DKU since 1947 단국대학교

Executive Summary

- Flash memory can achieve **50x endurance improvement by relaxing retention time with refresh** *[Cai+ ICCD '12]*
- *Problem*: **Refresh consumes the majority of endurance improvement**
- *Goal*: Reduce refresh overhead to increase flash memory lifetime
- *Key Observation*: **Refresh is unnecessary for *write-hot data***
- *Key Ideas of Write-hotness Aware Retention Management (WARM)*
  - **Physically partition write-hot pages and write-cold pages** within the flash drive
  - **Apply *different* policies** (garbage collection, wear-leveling, refresh) to each group
- *Key Results*
  - WARM w/o refresh **improves lifetime by 3.24x**
  - WARM w/ adaptive refresh **improves lifetime by 12.9x** (1.21x over refresh only)

Outline

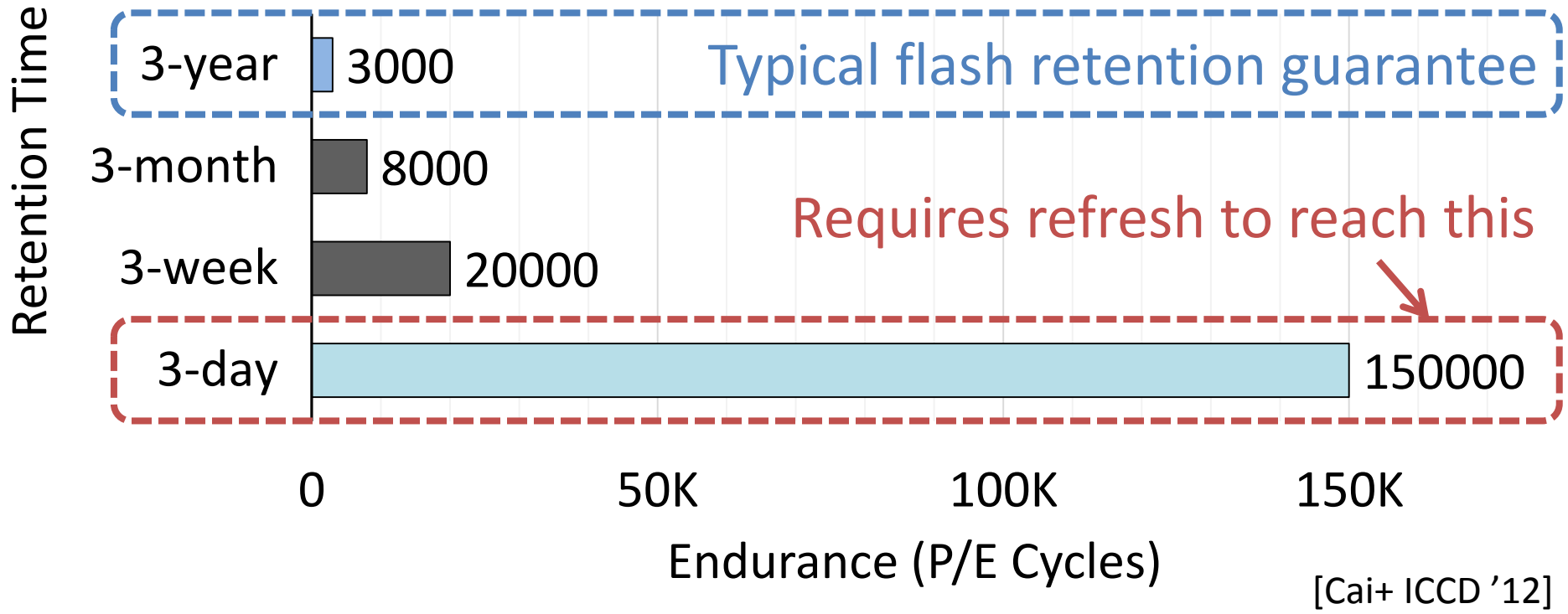
- *Problem and Goal*
- *Key Observations*
- *WARM: Write-hotness Aware Retention Management*
- *Results*
- *Conclusion*

Outline

- *Problem and Goal*
- *Key Observations*
- *WARM: Write-hotness Aware Retention Management*
- *Results*
- *Conclusion*

Retention Time Relaxation for Flash Memory

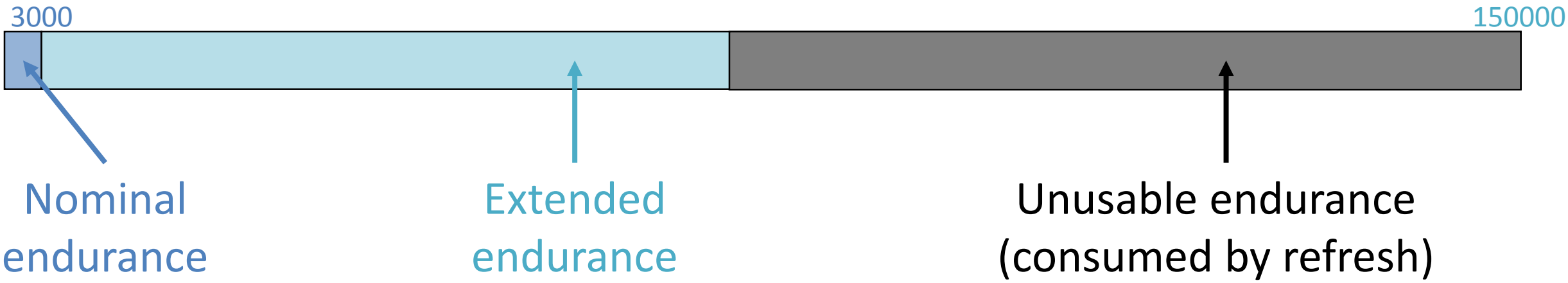
- Flash memory has limited *write endurance*
- *Retention time* significantly affects endurance
  - The duration for which flash memory correctly holds data



[Cai+ ICCD '12]

NAND Flash Refresh

- *Flash Correct and Refresh (FCR), Adaptive Rate FCR (ARFCR) [Cai+ ICCD '12]*



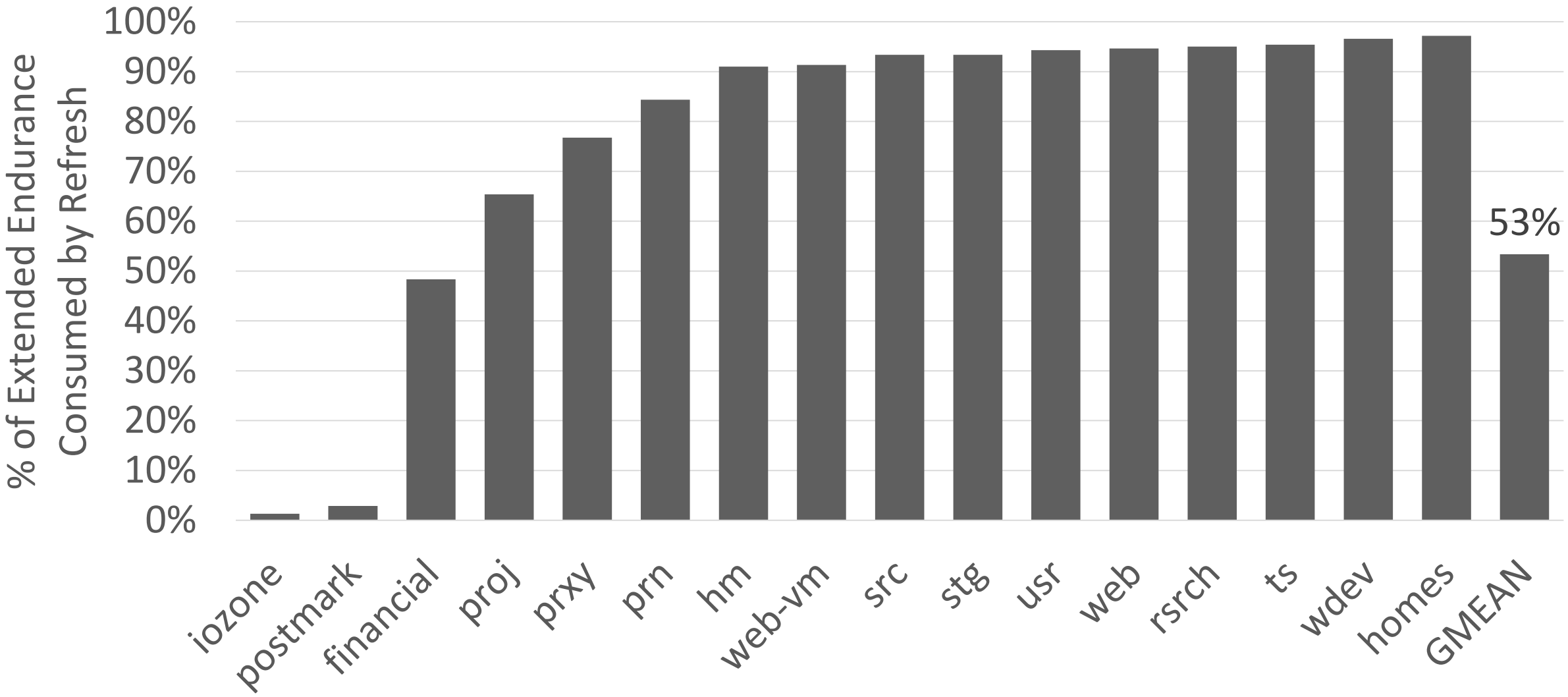
Problem: Flash refresh operations reduce extended lifetime

Goal: Reduce refresh overhead, improve flash lifetime

Outline

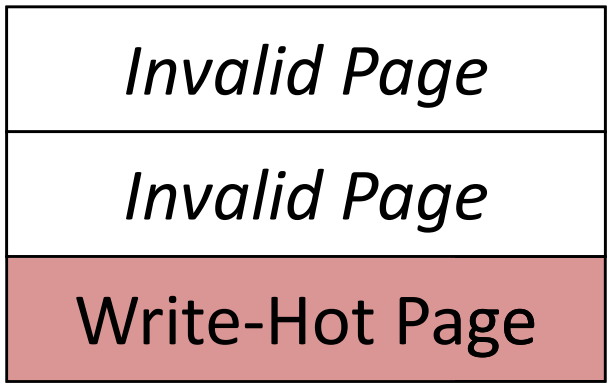
- *Problem and Goal*
- *Key Observations*
- *WARM: Write-hotness Aware Retention Management*
- *Results*
- *Conclusion*

Observation 1: Refresh Overhead is High



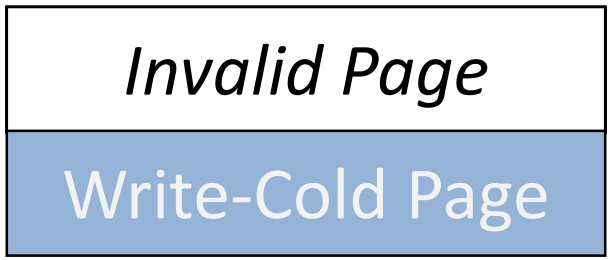
Observation 2: Write-Hot Pages Can Skip Refresh

Update



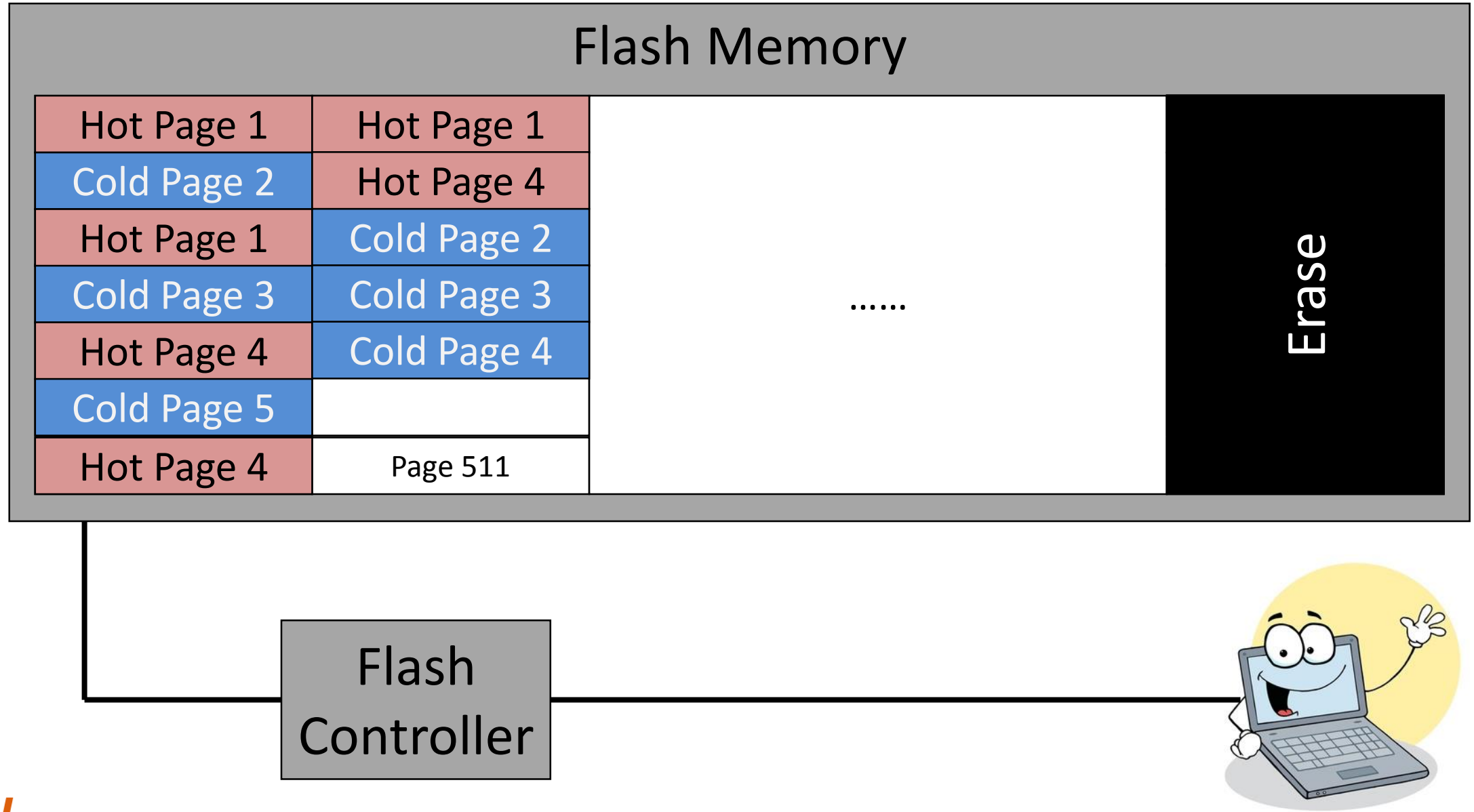
Skip Refresh

Retention Effect

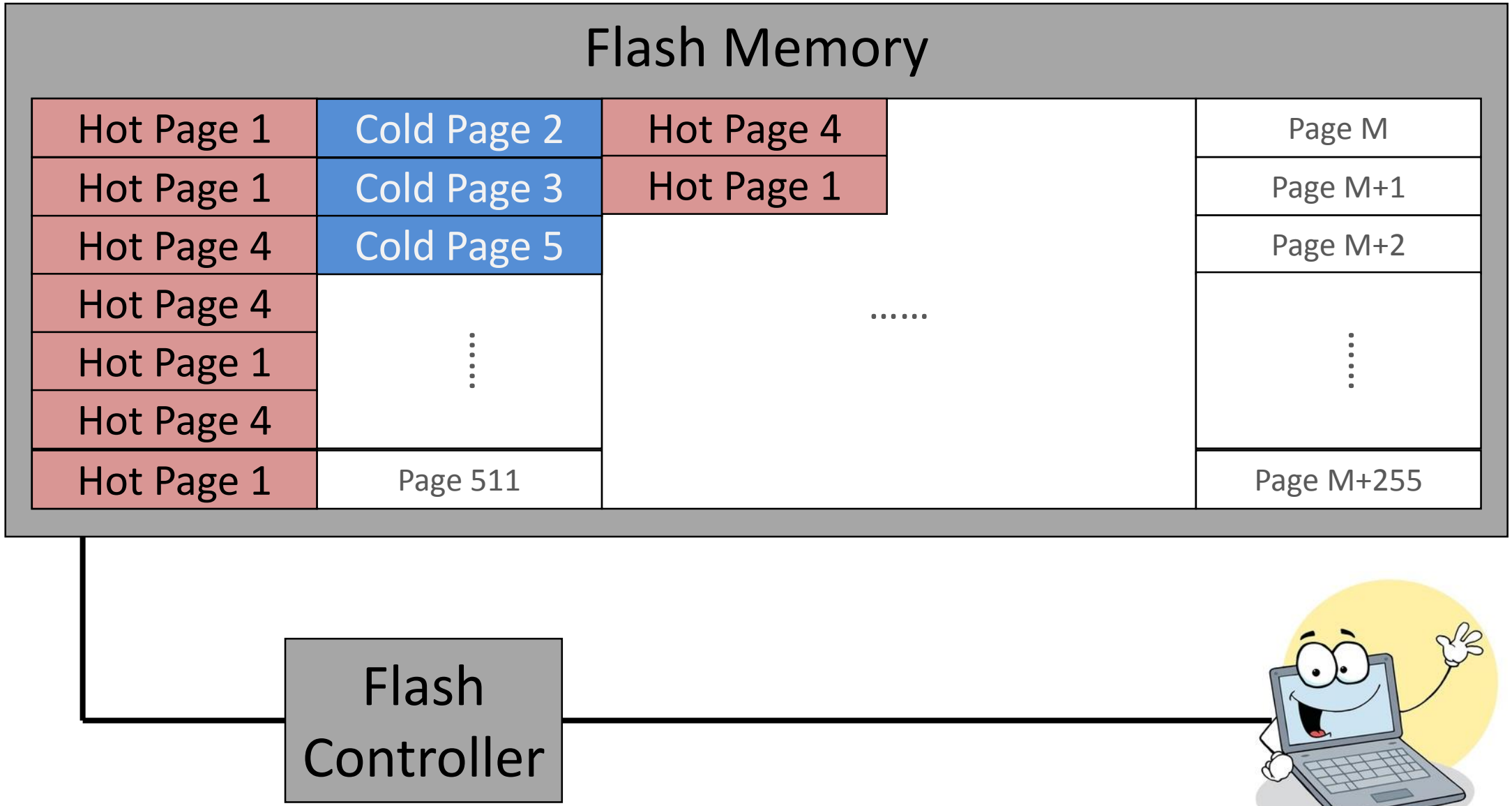


Need Refresh

Conventional Write-Hotness Oblivious Management



Key Idea: Write-Hotness Aware Management



Outline

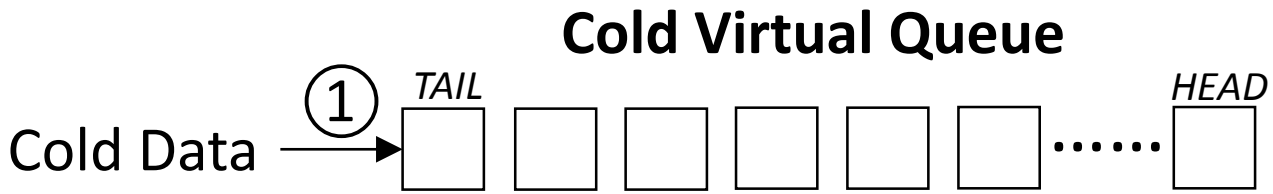
- *Problem and Goal*
- *Key Observations*
- *WARM: Write-hotness Aware Retention Management*
- *Results*
- *Conclusion*

WARM Overview

- Design Goal:
  - Relax retention time w/o refresh for write-hot data only

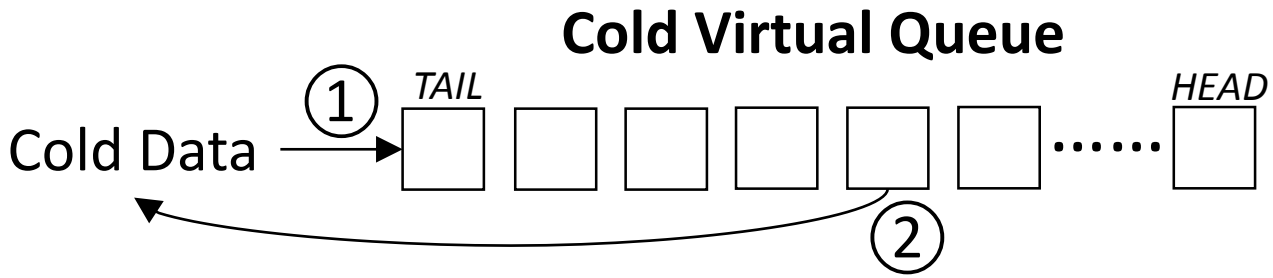
- WARM: Write-hotness Aware Retention Management
  - Write-hot/write-cold data partitioning algorithm
  - Write-hotness aware flash policies
    - *Partition write-hot and write-cold data into separate blocks*
    - *Skip refreshes for write-hot blocks*
    - *More efficient garbage collection and wear-leveling*

Write-Hot/Write-Cold Data Partitioning Algorithm



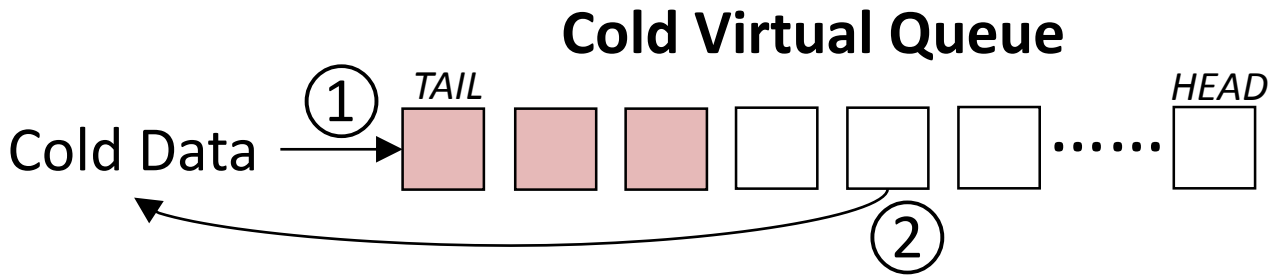
1. Initially, all data is cold and is stored in the cold virtual queue.

Write-Hot/Write-Cold Data Partitioning Algorithm



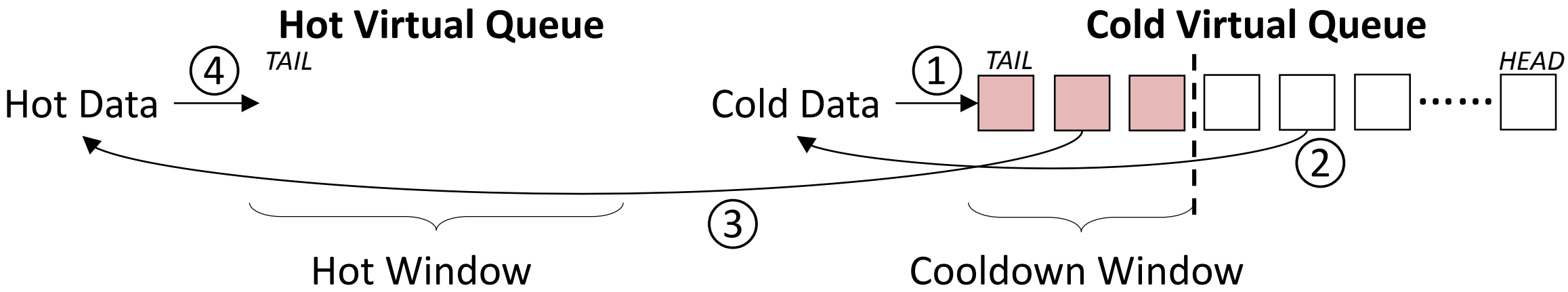
2. On a write operation, the data is pushed to the tail of the cold virtual queue.

Write-Hot/Write-Cold Data Partitioning Algorithm



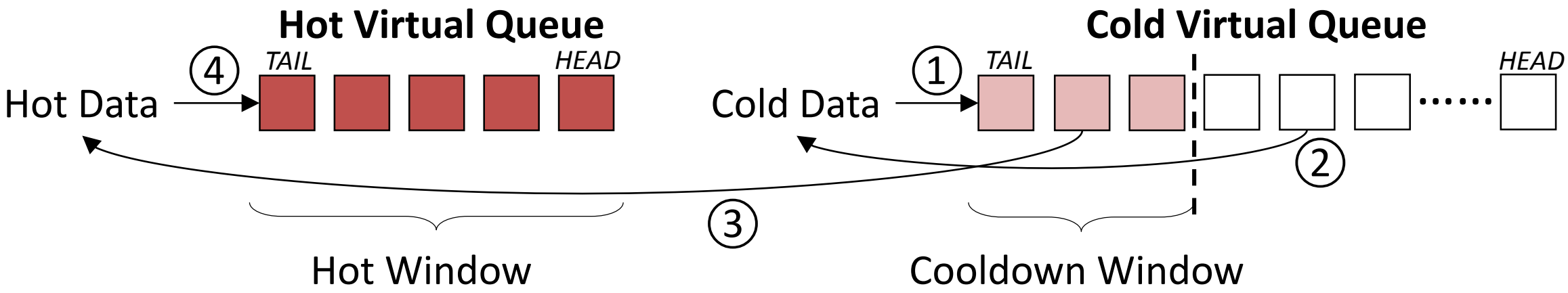
Recently-written data is at the tail of cold virtual queue.

Write-Hot/Write-Cold Data Partitioning Algorithm



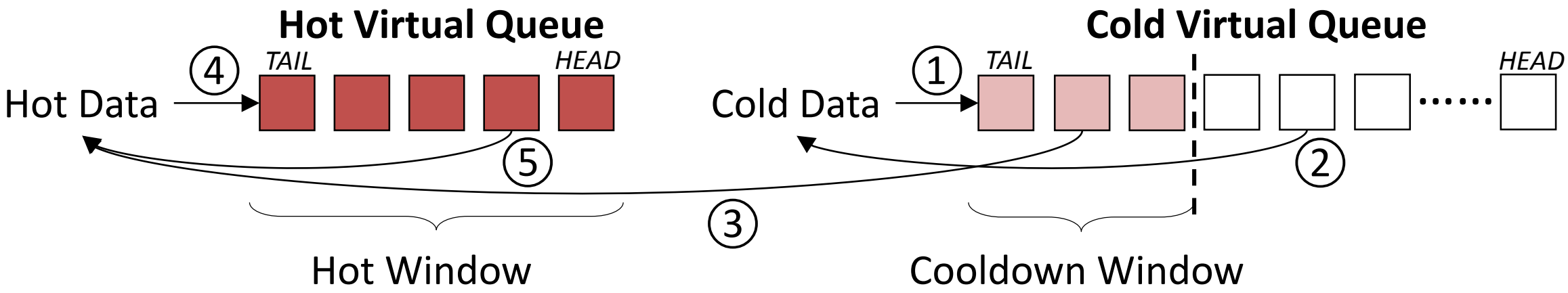
3, 4. On a write hit in the cooldown window,
the data is promoted to the hot virtual queue.

Write-Hot/Write-Cold Data Partitioning Algorithm



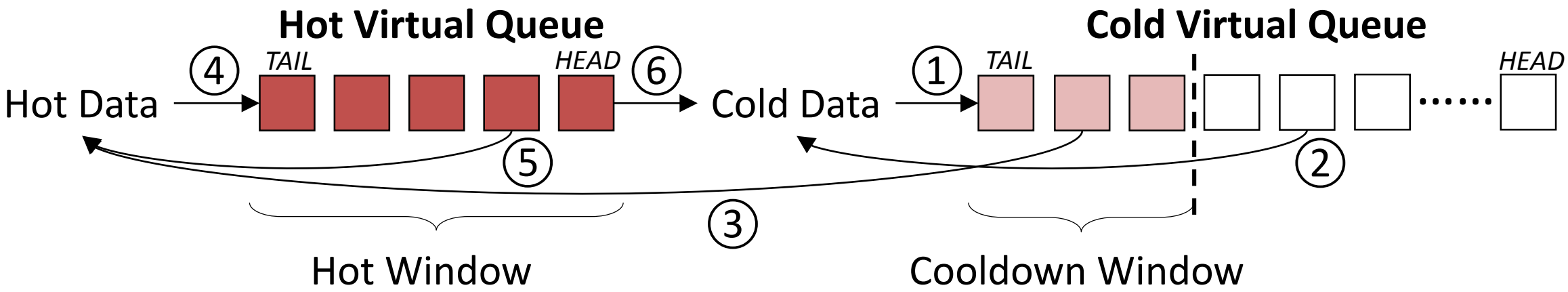
Data is sorted by write-hotness in the hot virtual queue.

Write-Hot/Write-Cold Data Partitioning Algorithm



5. On a write hit in hot virtual queue, the data is pushed to the tail.

Write-Hot/Write-Cold Data Partitioning Algorithm

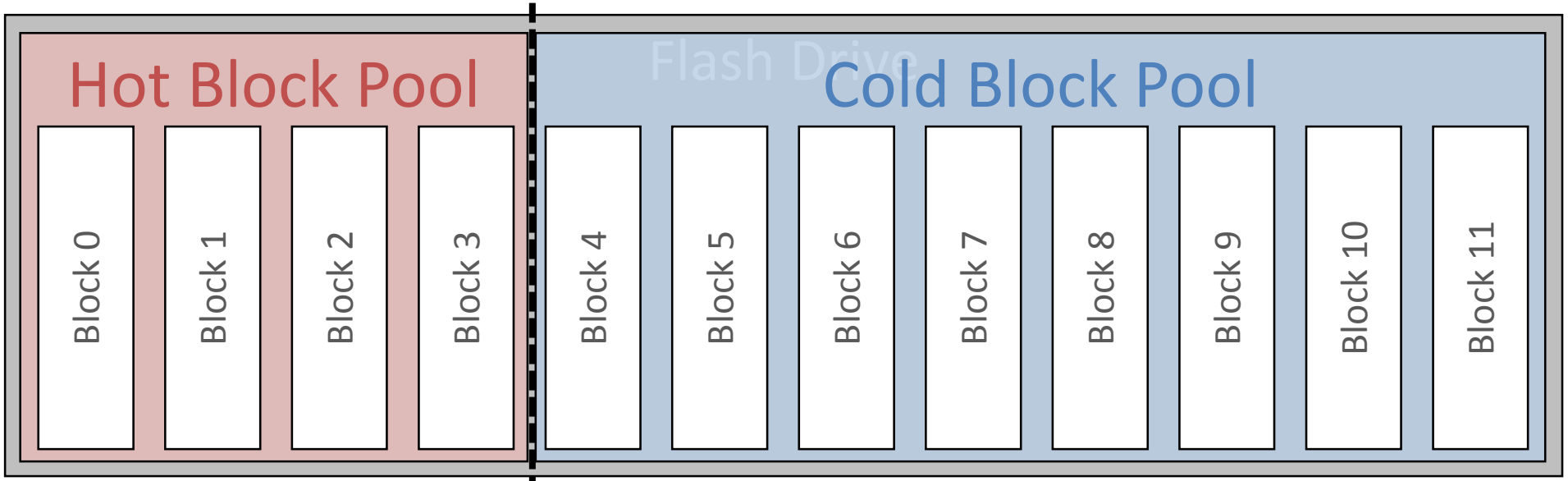


6. Unmodified hot data will be demoted to the cold virtual queue.

Conventional Flash Management Policies

- *Flash Translation Layer (FTL)*
  - Map data to erased blocks
  - Translate logical page number to physical page number
- *Garbage Collection*
  - Triggered before erasing a victim block
  - Remap all valid data on the victim block
- *Wear-leveling*
  - Triggered to balance wear-level among blocks

Write-Hotness Aware Flash Policies



- Write-hot data → naturally relaxed retention time

- Program in block order
- Garbage collect in block order
- All blocks naturally wear-leveled

- Write-cold data → lower write frequency, less wear-out

- Conventional garbage collection
- Conventional wear-leveling algorithm

Dynamically Sizing the Hot and Cold Block Pools

All blocks are divided between the hot and cold block pools

1. Find the maximum hot pool size
2. Reduce hot virtual queue size to maximize cold pool lifetime
3. Size the cooldown window to minimize ping-ponging of data between the two pools

Outline

- *Problem and Goal*
- *Key Observations*
- *WARM: Write-hotness Aware Retention Management*
- *Results*
- *Conclusion*

Methodology

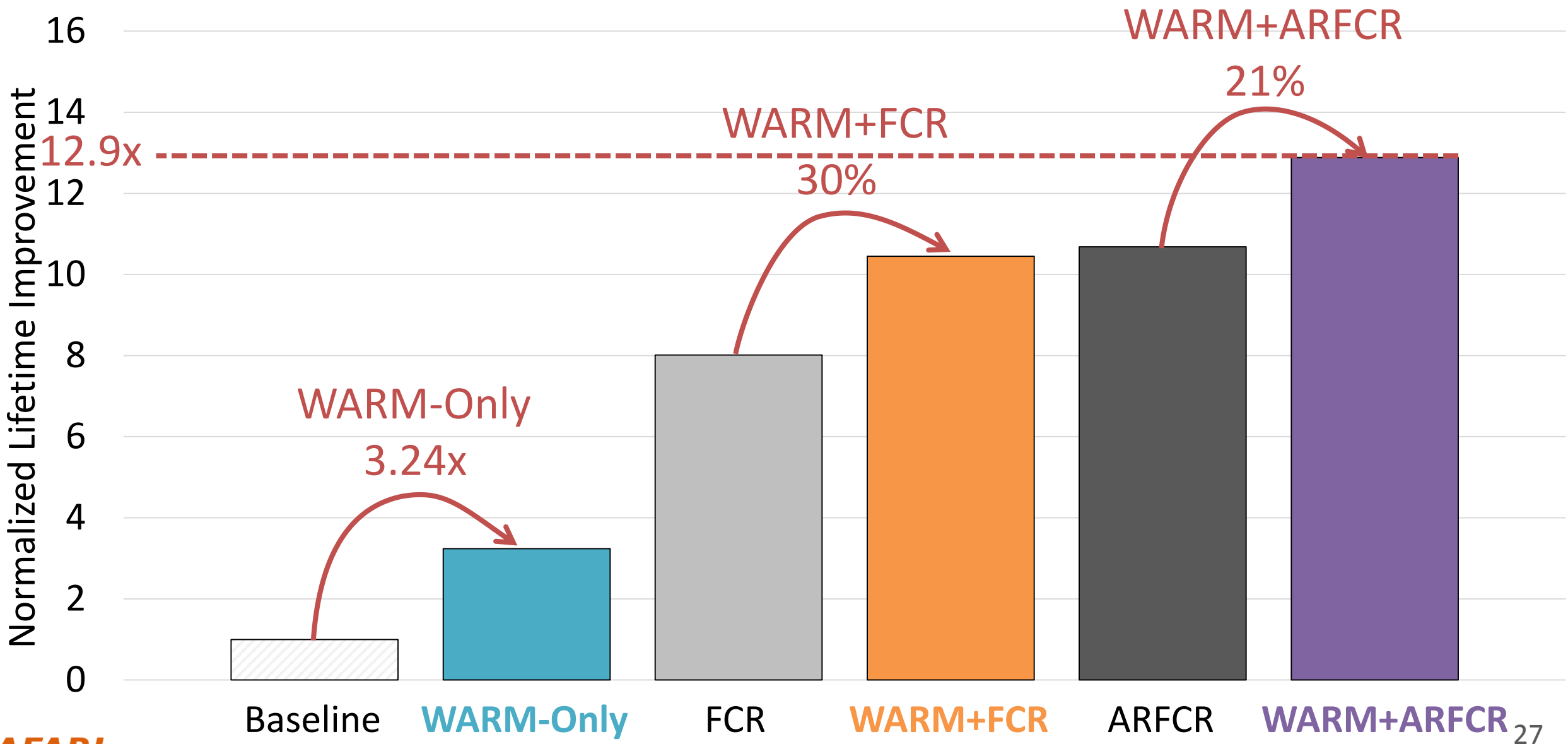
- *DiskSim 4.0 + SSD model*

Parameter	Value
Page read to register latency	25 µs
Page write from register latency	200 µs
Block erase latency	1.5 ms
Data bus latency	50 µs
Page/block size	8 KB/1 MB
Die/package size	8 GB/64 GB
Total capacity	256 GB
Over-provisioning	15%
Endurance for 3-year retention time	3,000 PEC
Endurance for 3-day retention time	150,000 PEC

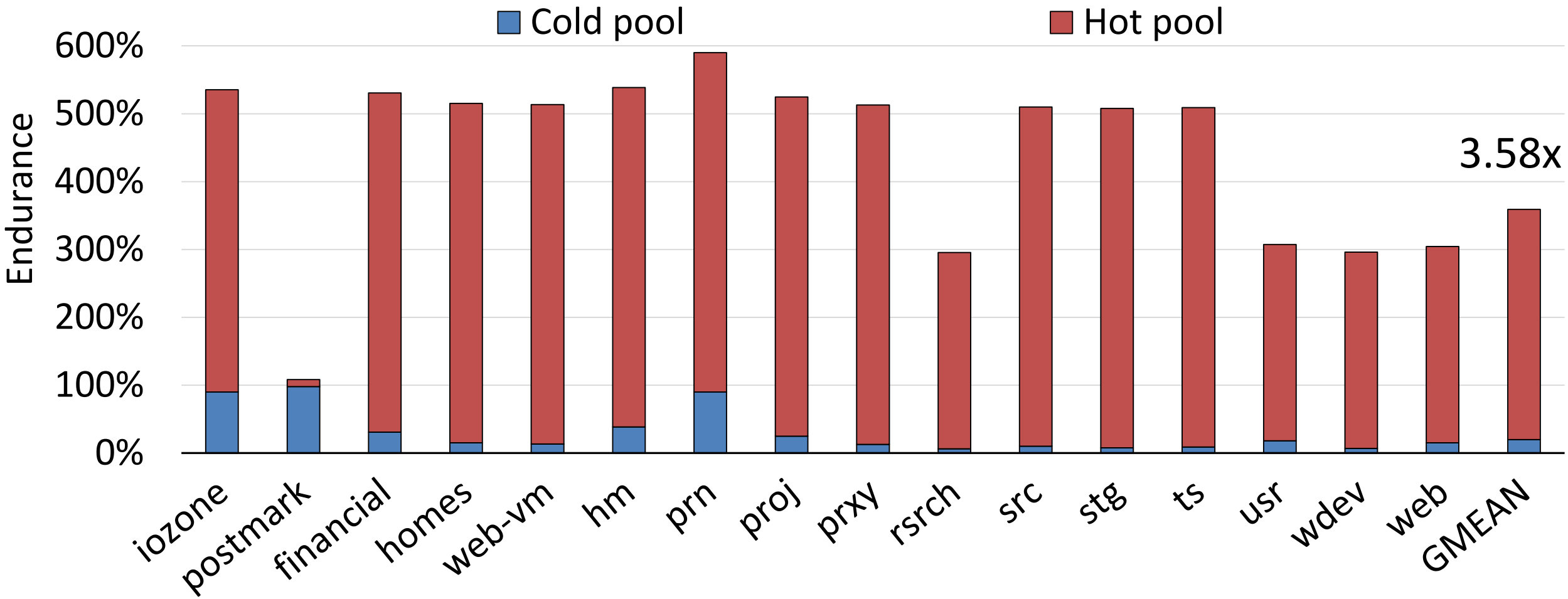
WARM Configurations

- *WARM-Only*
  - Relax retention time in hot block pool only
  - No refresh needed
- *WARM+FCR*
  - First apply **WARM-Only**
  - Then *also* relax retention time in cold block pool
  - Refresh cold blocks every 3 days
- *WARM+ARFCR*
  - Relax retention time in both hot and cold block pools
  - Adaptively increase the refresh frequency over time

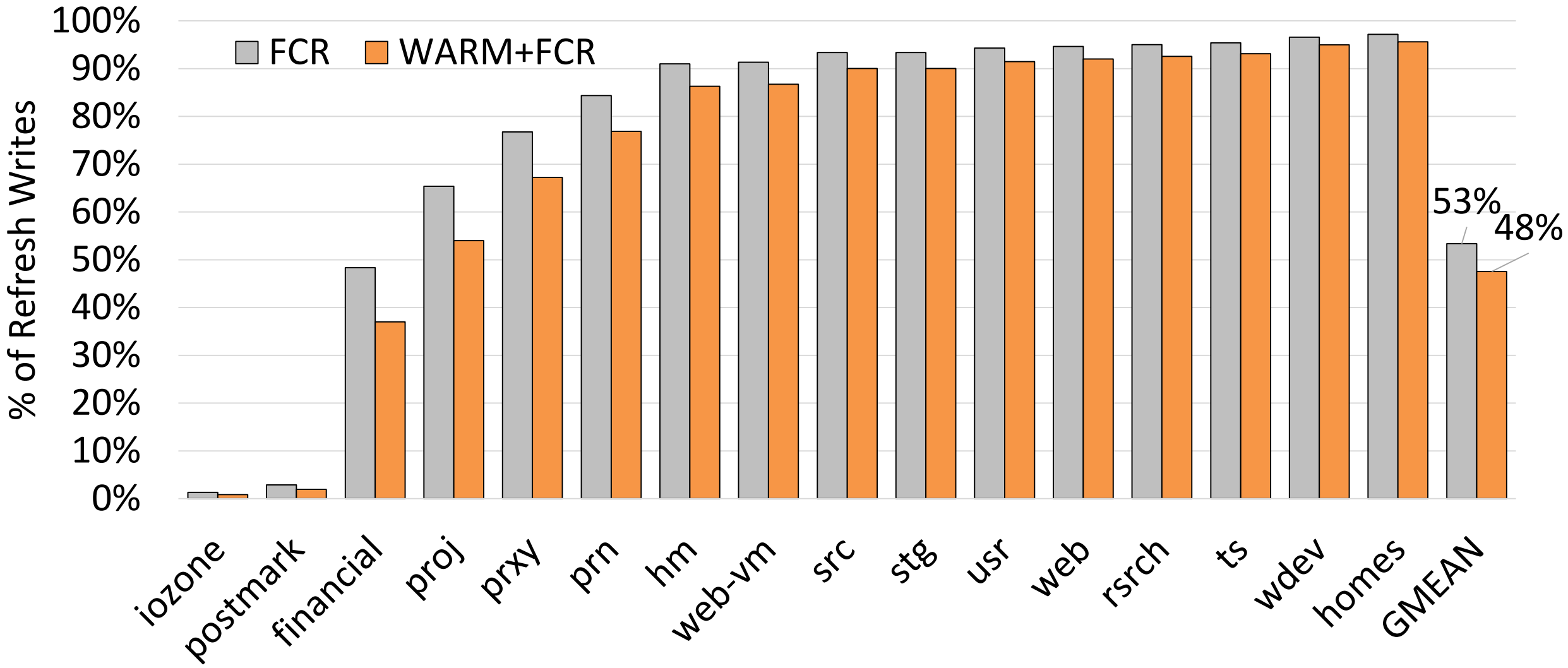
Flash Lifetime Improvements



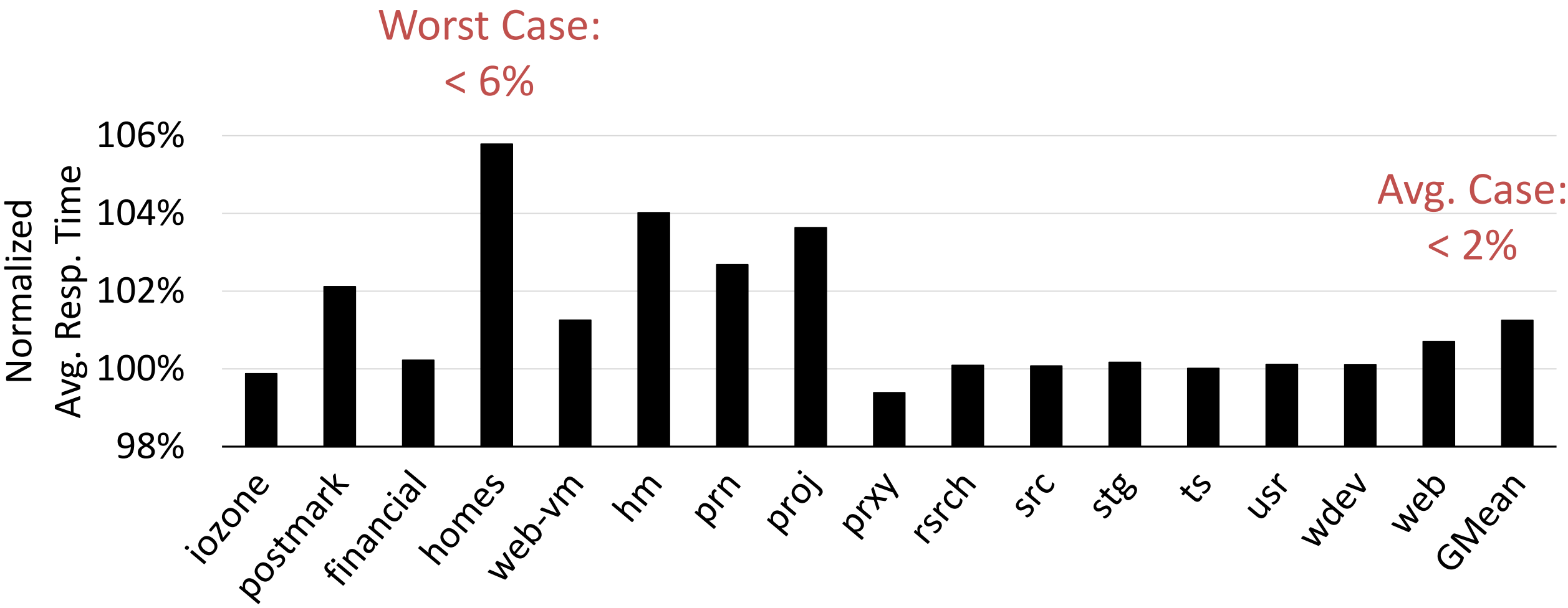
WARM-Only Endurance Improvement



WARM+FCR Refresh Operation Reduction



WARM Performance Impact



Other Results in the Paper

- *Breakdown of write frequency into host writes, garbage collection writes, refresh writes in the hot and cold block pools*
  - WARM reduces refresh writes significantly while having low garbage collection overhead
- *Sensitivity to different capacity over-provisioning amounts*
  - WARM improves flash lifetime more as over-provisioning increases
- *Sensitivity to different refresh intervals*
  - WARM improves flash lifetime more as refresh frequency increases

Outline

- *Problem and Goal*
- *Key Observations*
- *WARM: Write-hotness Aware Retention Management*
- *Results*
- *Conclusion*

Conclusion

- Flash memory can achieve **50x endurance improvement by relaxing retention time with refresh** *[Cai+ ICCD '12]*
- *Problem*: **Refresh consumes the majority of endurance improvement**
- *Goal*: Reduce refresh overhead to increase flash memory lifetime
- *Key Observation*: **Refresh is unnecessary for *write-hot data***
- *Key Ideas of Write-hotness Aware Retention Management (WARM)*
  - **Physically partition write-hot pages and write-cold pages** within the flash drive
  - **Apply *different* policies** (garbage collection, wear-leveling, refresh) to each group
- *Key Results*
  - WARM w/o refresh **improves lifetime by 3.24x**
  - WARM w/ adaptive refresh **improves lifetime by 12.9x** (1.21x over refresh only)

WARM

Improving NAND Flash Memory Lifetime with Write-hotness Aware Retention Management

***Yixin Luo**, Yu Cai, Saugata Ghose, Jongmoo Choi*, Onur Mutlu*

*Carnegie Mellon University, *Dankook University*

SAFARI

Carnegie Mellon

DANKOOK UNIVERSITY DKU since 1947 단국대학교

Backup Slides

Related Work: Retention Time Relaxation

- Perform *periodic refresh* on data to relax retention time [Cai+ ICCD '12, Cai+ ITJ '13, Liu+ DAC '13, Pan+ HPCA '12]
  - Fixed-frequency refresh (e.g., FCR)
  - Adaptive refresh (e.g., ARFCR): incrementally increase refresh freq.
  - Incurs a **high overhead**, since block-level erase/rewrite required
  - WARM can work **alongside** periodic refresh
- Refresh using rewriting codes [Li+ ISIT '14]
  - Avoids block-level erasure
  - Adds **complex encoding/decoding circuitry** into flash memory

Related Work: Hot/Cold Data Separation in FTLs

- Mechanisms with ***statically-sized*** windows/bins for partitioning
  - Multi-level hash tables to improve FTL latency [Lee+ TCE '09, Wu+ ICCAD '06]
  - Sorted tree for wear-leveling [Chang SAC '07]
  - Log buffer migration for garbage collection [Lee+ OSR '08]
  - Multiple static queues for garbage collection [Chang+ RTAS '02, Chiang SPE '99, Jung CSA '13]
  - Static window sizing **bad for WARM**
    - *Number of write-hot pages changes over time*
    - Undersized: **reduced benefits**
    - *Oversized: **data loss** of cold pages incorrectly in hot page window*

Related Work: Hot/Cold Data Separation in FTLs

- Estimating page update frequency for ***dynamic*** partitioning
  - Using most recent re-reference distance for garbage collection [Stoica VLDB '13] or for write buffer locality [Wu+ MSST '10]
  - Using multiple Bloom filters for garbage collection [Park MSST '11]
  - Prone to **false positives**: increased migration for WARM
  - Reverse translation to logical page no. consumes **high overhead**
- Placing write-hot data in ***worn-out pages*** [Huang+ EuroSys '14]
  - Assumes SSD w/o refresh
  - Benefits **limited** by number of worn-out pages in SSD
  - Hot data pool size cannot be dynamically adjusted

Related Work: Non-FTL Hot/Cold Data Separation

- These works all use multiple ***statically-sized*** queues
  - Reference counting for garbage collection [Joao+ ISCA '09]
  - Cache replacement algorithms [Johnson+ VLDB '94, Megiddo+ FAST '03, Zhou+ ATC '01]
- Static window sizing **bad for WARM**
  - Number of write-hot pages changes over time
  - Undersized: **reduced benefits**
  - Oversized: **data loss** of cold pages incorrectly in hot page window

Other Work by SAFARI on Flash Memory

- J. Meza, Q. Wu, S. Kumar, and O. Mutlu. *A Large-Scale Study of Flash Memory Errors in the Field*, SIGMETRICS 2015.
- Y. Cai, Y. Luo, S. Ghose, E. F. Haratsch, K. Mai, O. Mutlu. *Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation*, DSN 2015.
- Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, O. Mutlu. *Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery*, HPCA 2015.
- Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, O. Unsal, A. Cristal, K. Mai. *Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories*, SIGMETRICS 2014.
- Y. Cai, O. Mutlu, E. F. Haratsch, K. Mai. *Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation*, ICCD 2013.
- Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, K. Mai. *Error Analysis and Retention-Aware Error Management for NAND Flash Memory*, Intel Technology Jrnl. (ITJ), Vol. 17, No. 1, May 2013.
- Y. Cai, E. F. Haratsch, O. Mutlu, K. Mai. *Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis and Modeling*, DATE 2013.
- Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, K. Mai. *Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime*, ICCD 2012.
- Y. Cai, E. F. Haratsch, O. Mutlu, K. Mai. *Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis*, DATE 2012.

References

- [Cai+ ICCD '12] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, K. Mai. ***Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime***, ICCD 2012.
- [Cai+ ITJ '13] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, K. Mai. ***Error Analysis and Retention-Aware Error Management for NAND Flash Memory***, Intel Technology Jrnl. (ITJ), Vol. 17, No. 1, May 2013.
- [Chang SAC '07] L.-P. Chang. ***On Efficient Wear Leveling for Large-Scale Flash-Memory Storage Systems***, SAC 2007.
- [Chang+ RTAS '02] L.-P. Chang, T.-W. Kuo. ***An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems***, RTAS 2002.
- [Chiang SPE '99] M.-L. Chiang, P. C. H. Lee, R.-C. Chang. ***Using Data Clustering to Improve Cleaning Performance for Flash Memory***, Software: Practice & Experience (SPE), 1999.
- [Huang+ EuroSys '14] P. Huang, G. Wu, X. He, W. Xiao. ***An Aggressive Worn-out Flash Block Management Scheme to Alleviate SSD Performance Degradation***, EuroSys 2014.
- [Joao+ ISCA '09] J. A. Joao, O. Mutlu, Y. N. Patt. ***Flexible Reference-Counting-Based Hardware Acceleration for Garbage Collection***, ISCA 2009.
- [Johnson+ VLDB '94] T. Johnson, D. Shasha. ***2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm***, VLDB 1994.
- [Jung CSA '13] T. Jung, Y. Lee, J. Woo, I. Shin. ***Double Hot/Cold Clustering for Solid State Drives***, CSA 2013.

References

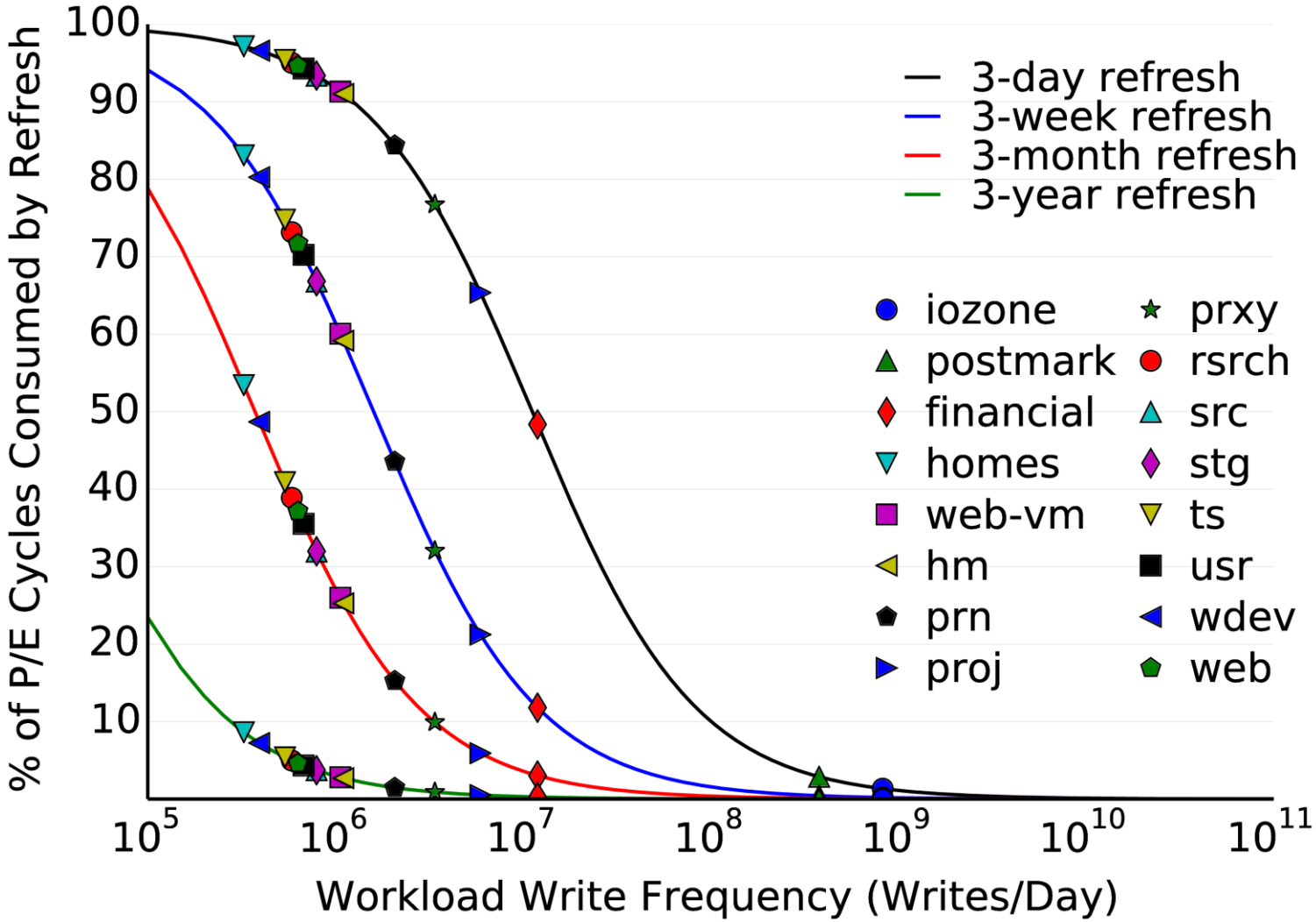
- [Lee+ OSR '08] S. Lee, D. Shin, Y.-J. Kim, J. Kim. ***LAST: Locality-Aware Sector Translation for NAND Flash Memory-Based Storage Systems***, ACM SIGOPS Operating Systems Review (OSR), 2008.
- [Lee+ TCE '09] H.-S. Lee, H.-S. Yun, D.-H. Lee. ***HFTL: Hybrid Flash Translation Layer Based on Hot Data Identification for Flash Memory***, IEEE Trans. Consumer Electronics (TCE), 2009.
- [Li+ ISIT '14] Y. Li, A. Jiang, J. Bruck. ***Error Correction and Partial Information Rewriting for Flash Memories***, ISIT 2014.
- [Liu+ DAC '13] R.-S. Liu, C.-L. Yang, C.-H. Li, G.-Y. Chen. ***DuraCache: A Durable SSD Cache Using MLC NAND Flash***, DAC 2013.
- [Megiddo+ FAST '03] N. Megiddo, D. S. Modha. ***ARC: A Self-Tuning, Low Overhead Replacement Cache***, FAST 2003.
- [Pan+ HPCA '12] Y. Pan, G. Dong, Q. Wu, T. Zhang. ***Quasi-Nonvolatile SSD: Trading Flash Memory Nonvolatility to Improve Storage System Performance for Enterprise Applications***, HPCA 2012.
- [Park MSST '11] D. Park, D. H. Du. ***Hot Data Identification for Flash-Based Storage Systems Using Multiple Bloom Filters***, MSST 2011.
- [Stoica VLDB '13] R. Stoica and A. Ailamaki. ***Improving Flash Write Performance by Using Update Frequency***, VLDB 2013.
- [Wu+ ICCAD '06] C.-H. Wu, T.-W. Kuo. ***An Adaptive Two-Level Management for the Flash Translation Layer in Embedded Systems***, ICCAD 2006.
- [Wu+ MSST '10] G. Wu, B. Eckart, X. He. ***BPAC: An Adaptive Write Buffer Management Scheme for Flash-based Solid State Drives***, MSST 2010.
- [Zhou+ ATC '01] Y. Zhou, J. Philbin, K. Li. ***The Multi-Queue Replacement Algorithm for Second Level Buffer Caches***, USENIX ATC 2001.

Workloads Studied

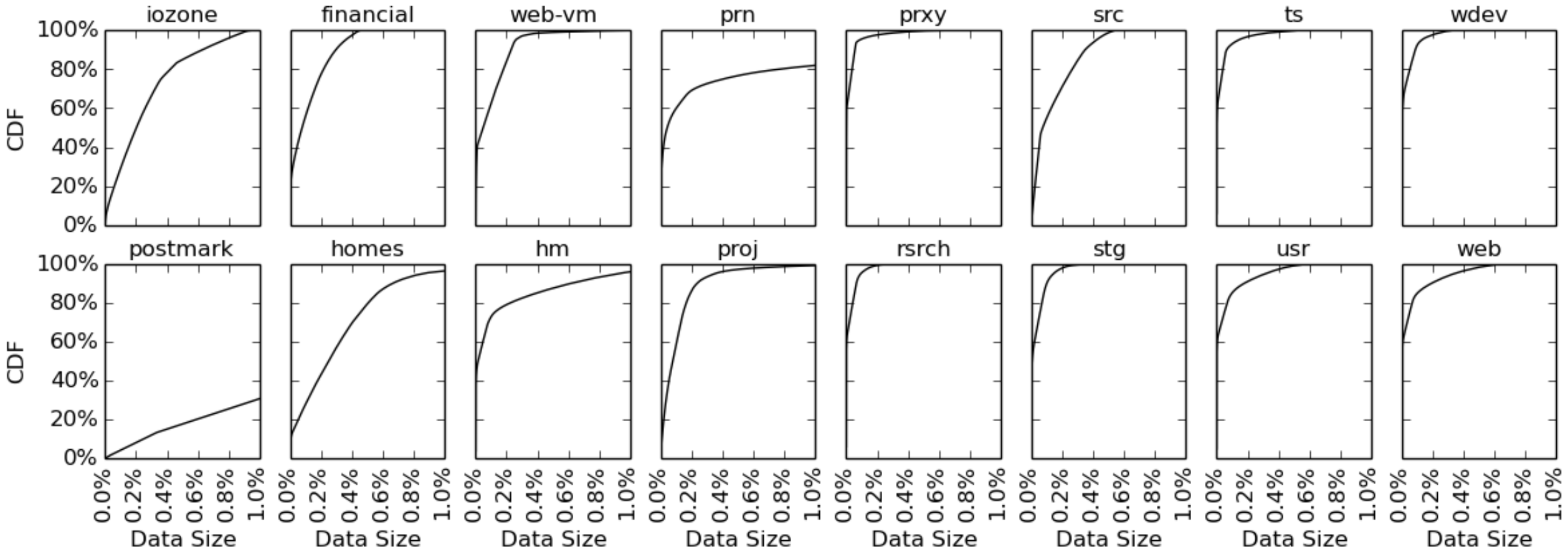
Synthetic Workloads							
Trace	**Source**	**Length**	**Description**	**Trace**	**Source**	**Length**	**Description**
iozone	IOzone	16 min	File system benchmark	postmark	Postmark	8.3 min	File system benchmark

Real-World Workloads							
Trace	**Source**	**Length**	**Description**	**Trace**	**Source**	**Length**	**Description**
financial	UMass	1 day	Online transaction processing	rsrch	MSR	7 days	Research projects
homes	FIU	21 days	Research group activities	src	MSR	7 days	Source control
web-vm	FIU	21 days	Web mail proxy server	stg	MSR	7 days	Web staging
hm	MSR	7 days	Hardware monitoring	ts	MSR	7 days	Terminal server
prn	MSR	7 days	Print server	usr	MSR	7 days	User home directories
proj	MSR	7 days	Project directories	wdev	MSR	7 days	Test web server
prxy	MSR	7 days	Firewall/web proxy	web	MSR	7 days	Web/SQL server

Refresh Overhead vs. Write Frequency

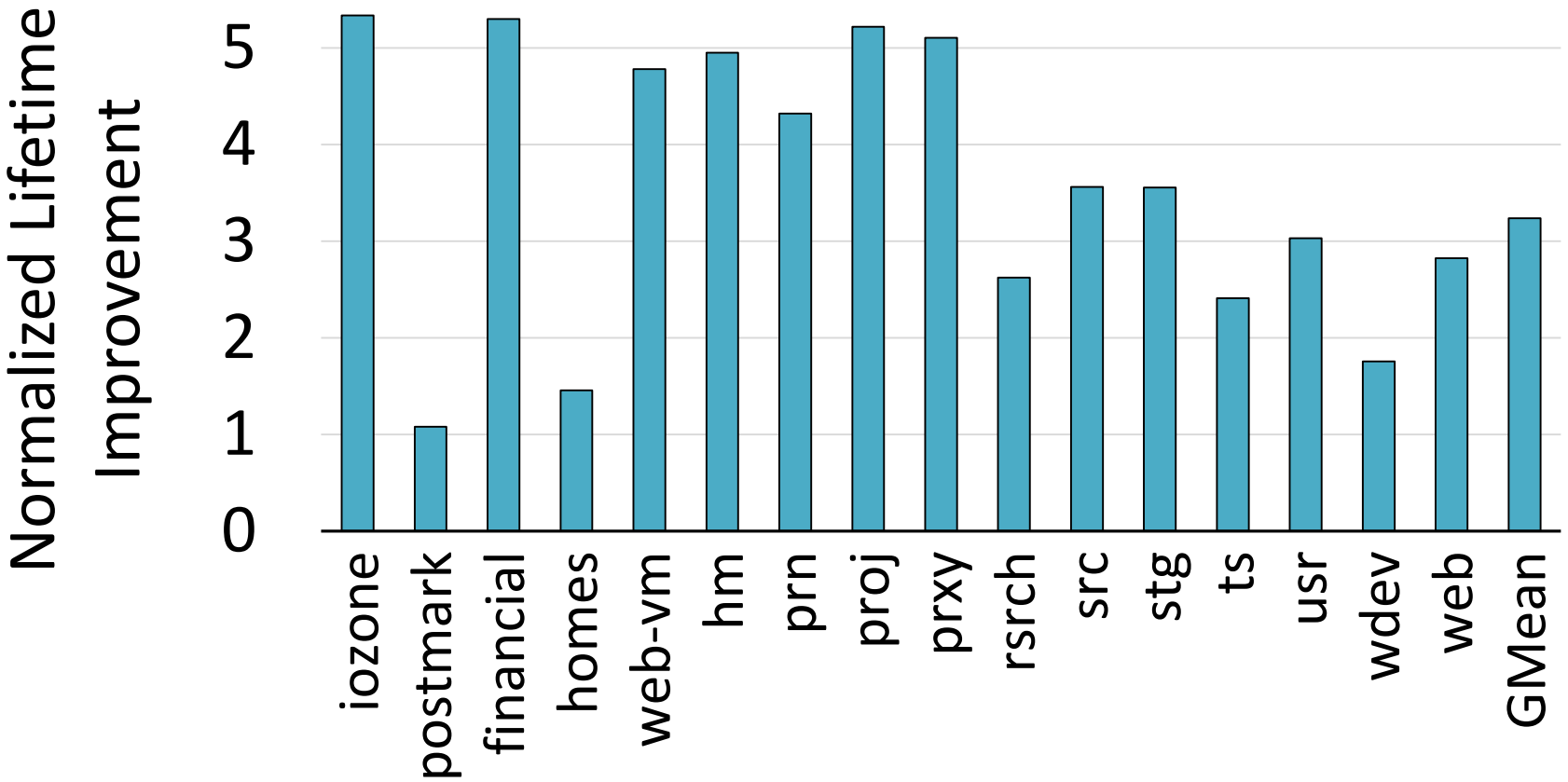


Highly-Skewed Distribution of Write Activity

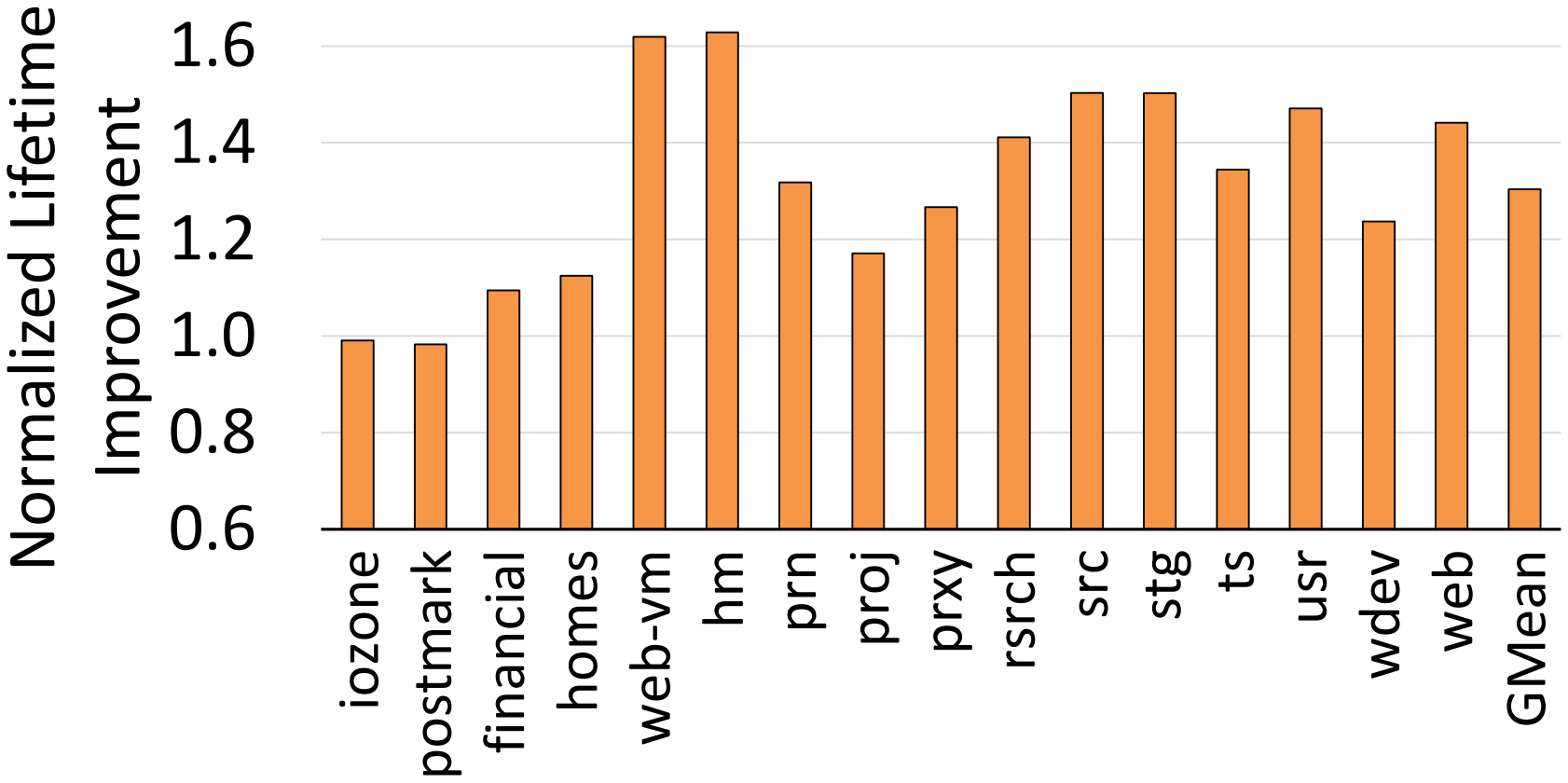


Small amount of write-hot data generates large fraction of writes.

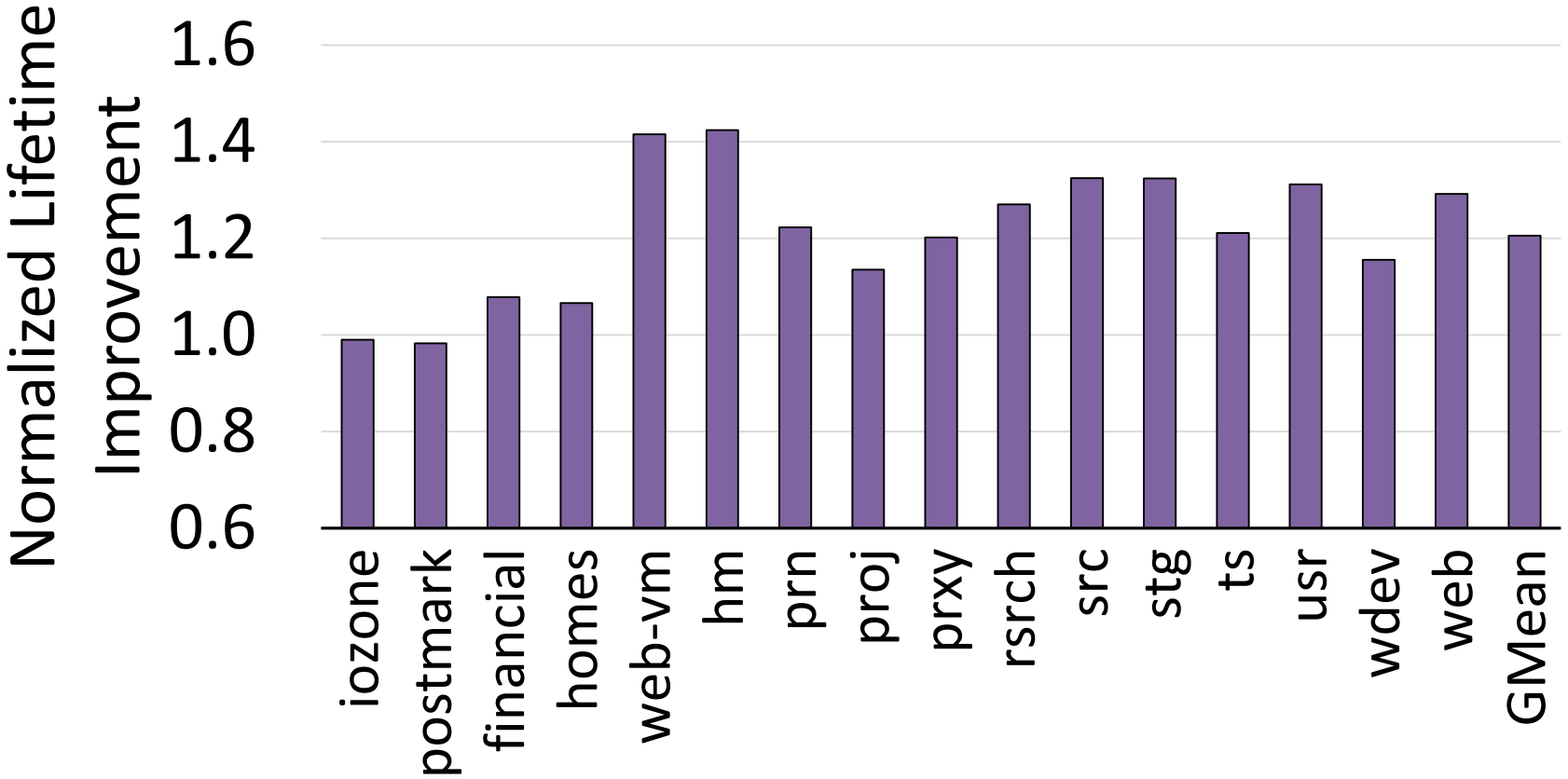
WARM-Only vs. Baseline



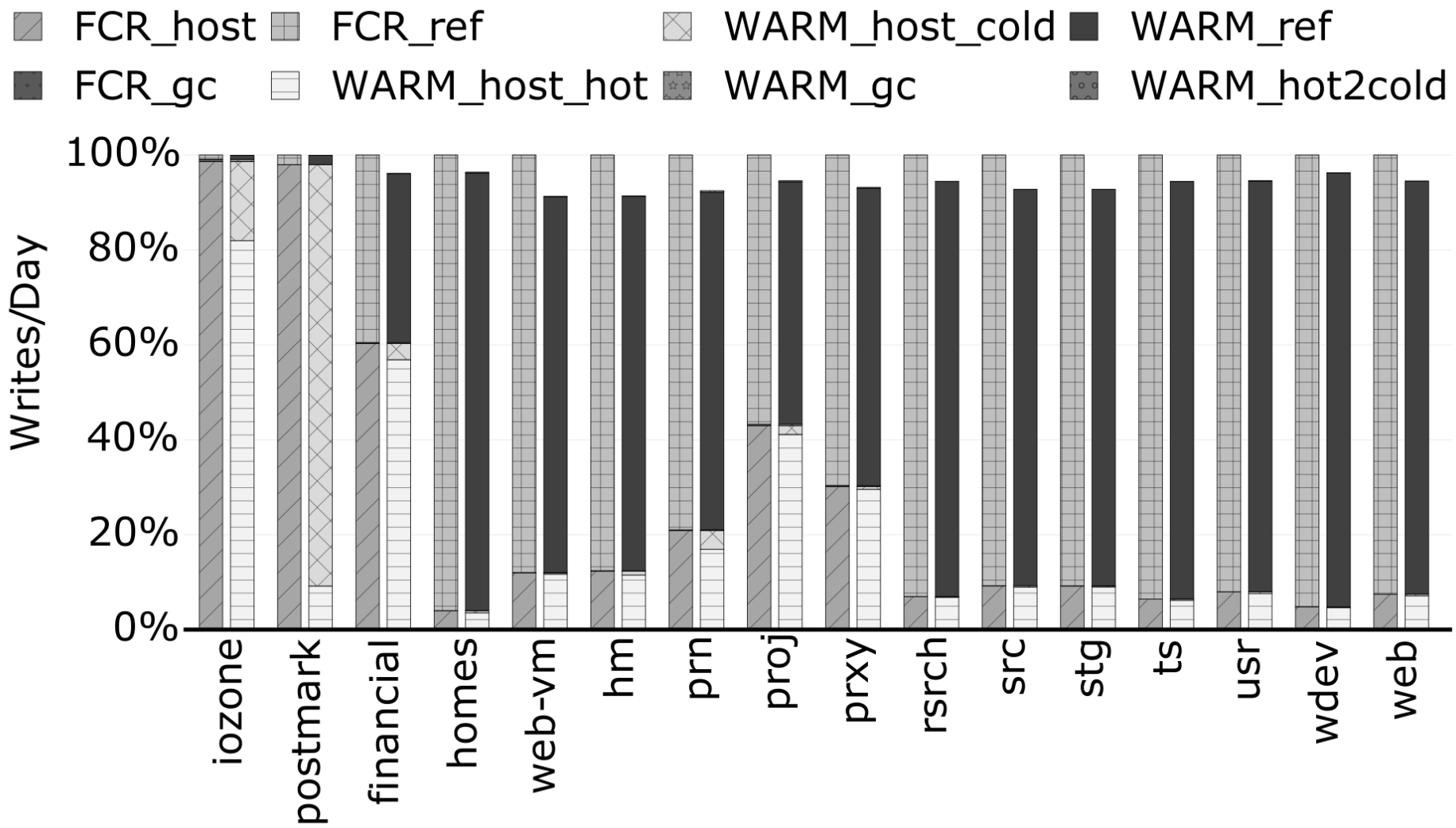
WARM+FCR vs. FCR-Only



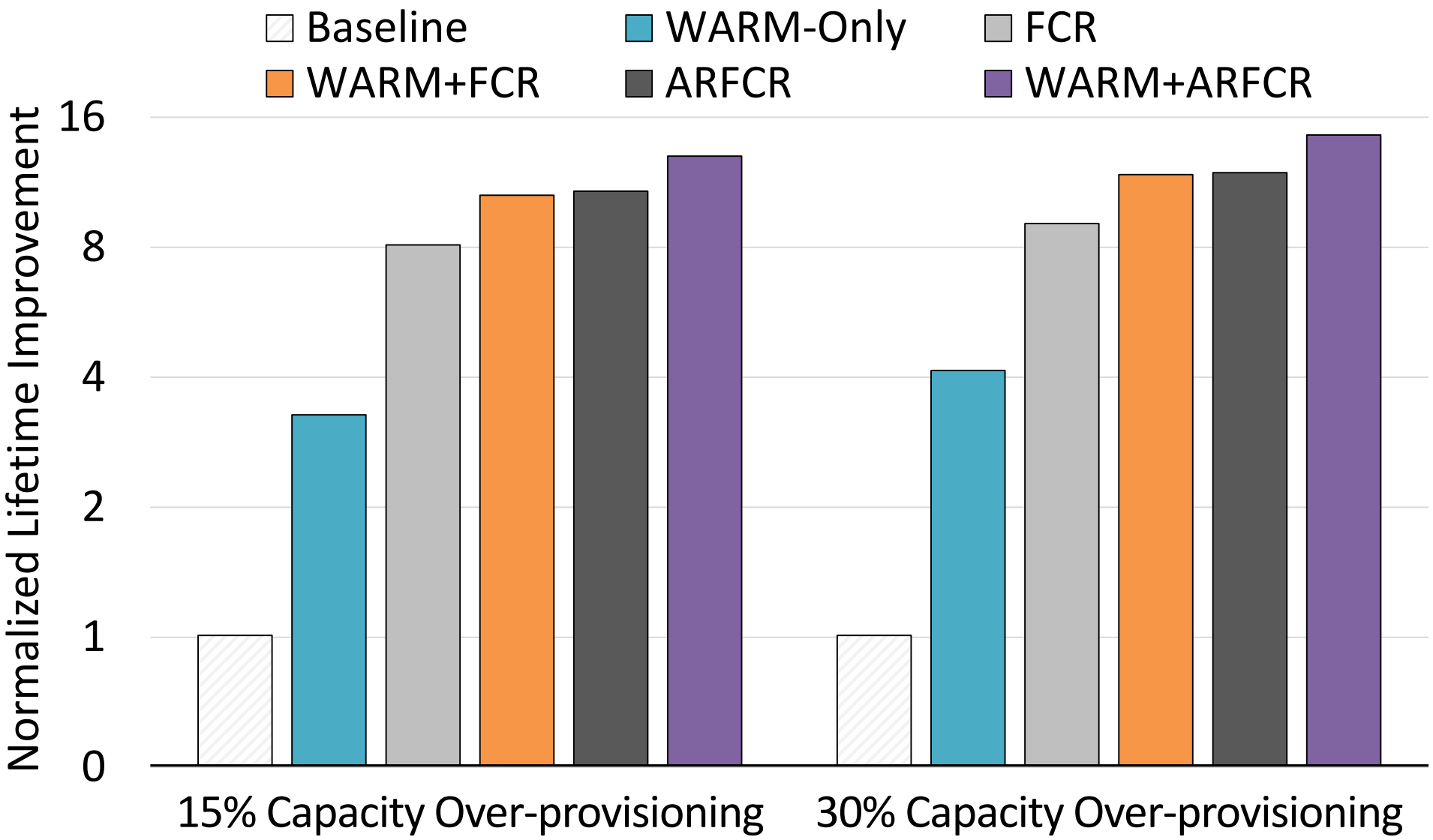
WARM+ARFCR vs. ARFCR-Only



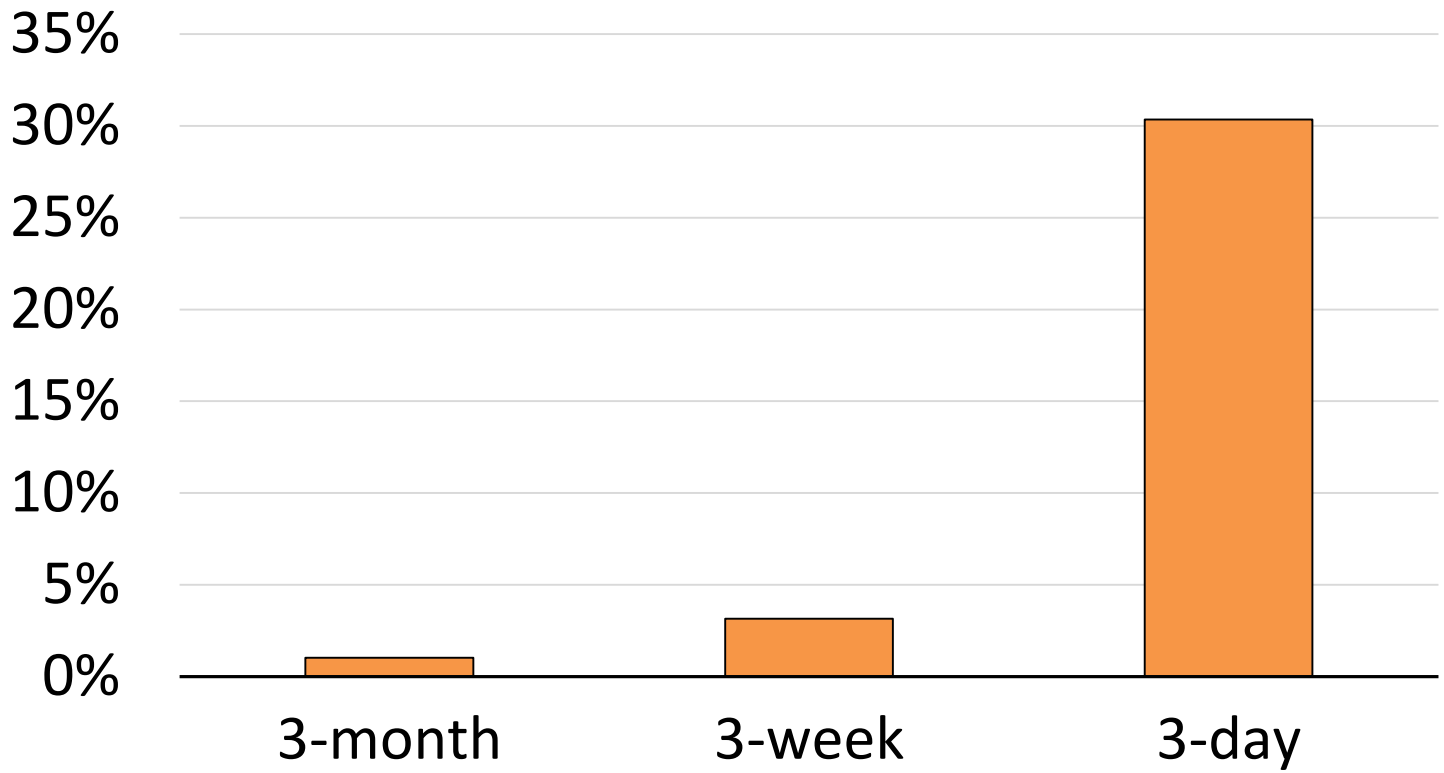
Breakdown of Writes



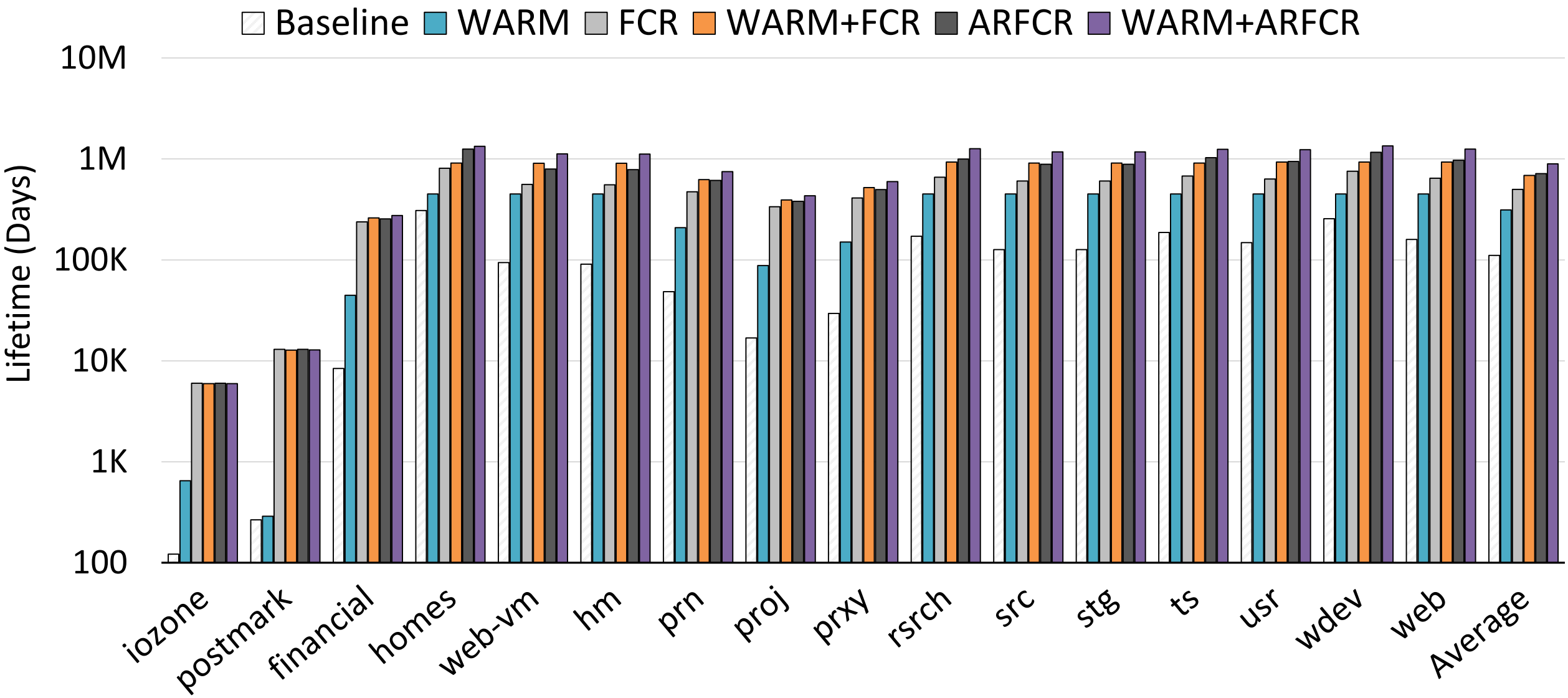
Sensitivity to Capacity Over-Provisioning



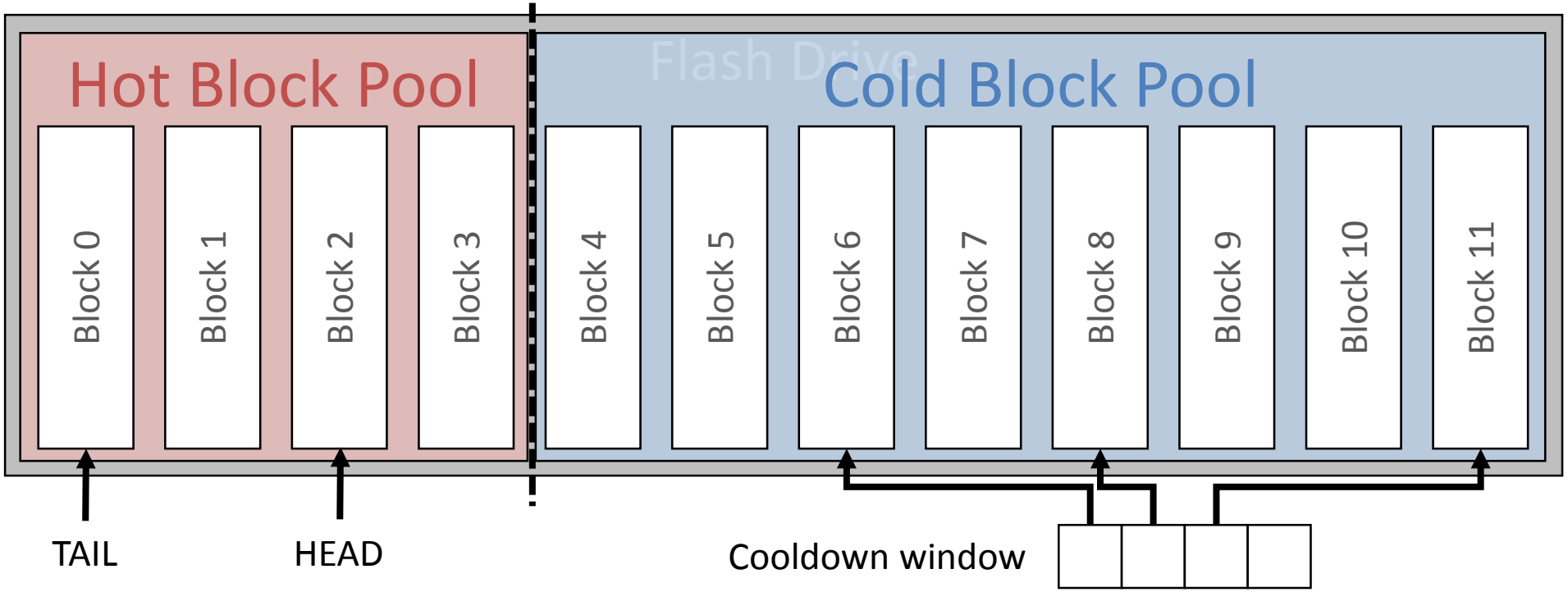
Sensitivity to Refresh Frequency



Lifetime Improvement from WARM



WARM Flash Management Policies



- *Dynamic hot and cold block pool partitioning*
  - Cold pool lifetime = $\frac{Cold\ pool\ endurance\ capacity}{Cold\ write\ frequency} \propto \frac{Cold\ pool\ size}{Cold\ write\ frequency}$
- *Cooldown window size tuning*
  - Minimize unnecessary promotion to hot block pool

Revisit WARM Design Goals

Write-hot/write-cold data partition algorithm

Goal 1: Partition write-hot and write-cold data ✓

Goal 2: Quickly adapt to workload behavior ✓

Flash management policies

Goal 3: Apply different management policies to improve flash lifetime ✓

- Skip refreshes in hot block pool
- Increase garbage collection efficiency

Goal 4: Low implementation and performance overhead ✓

- 4 counters and ~1KB storage overhead