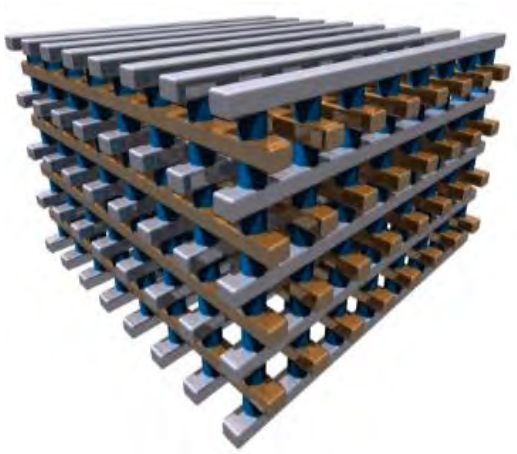
Ouroboros Wear-leveling: A Two-level Hierarchical Wear-leveling Model for NVRAM

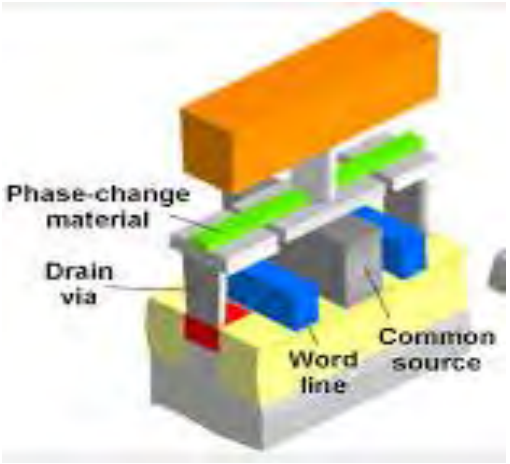
Qingyue Liu
Peter Varman
ECE Department, Rice University

May 18, 2017

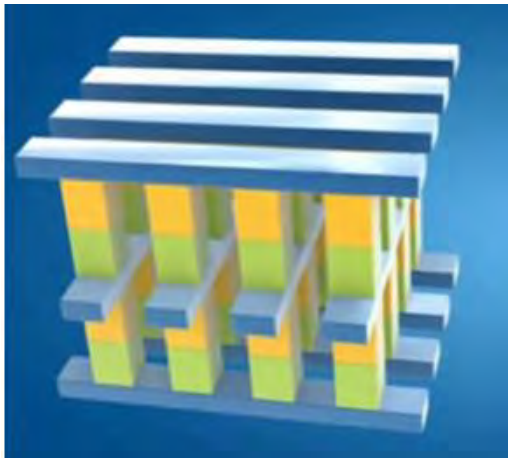
New Challenges for New Technologies



RRAM



PCM



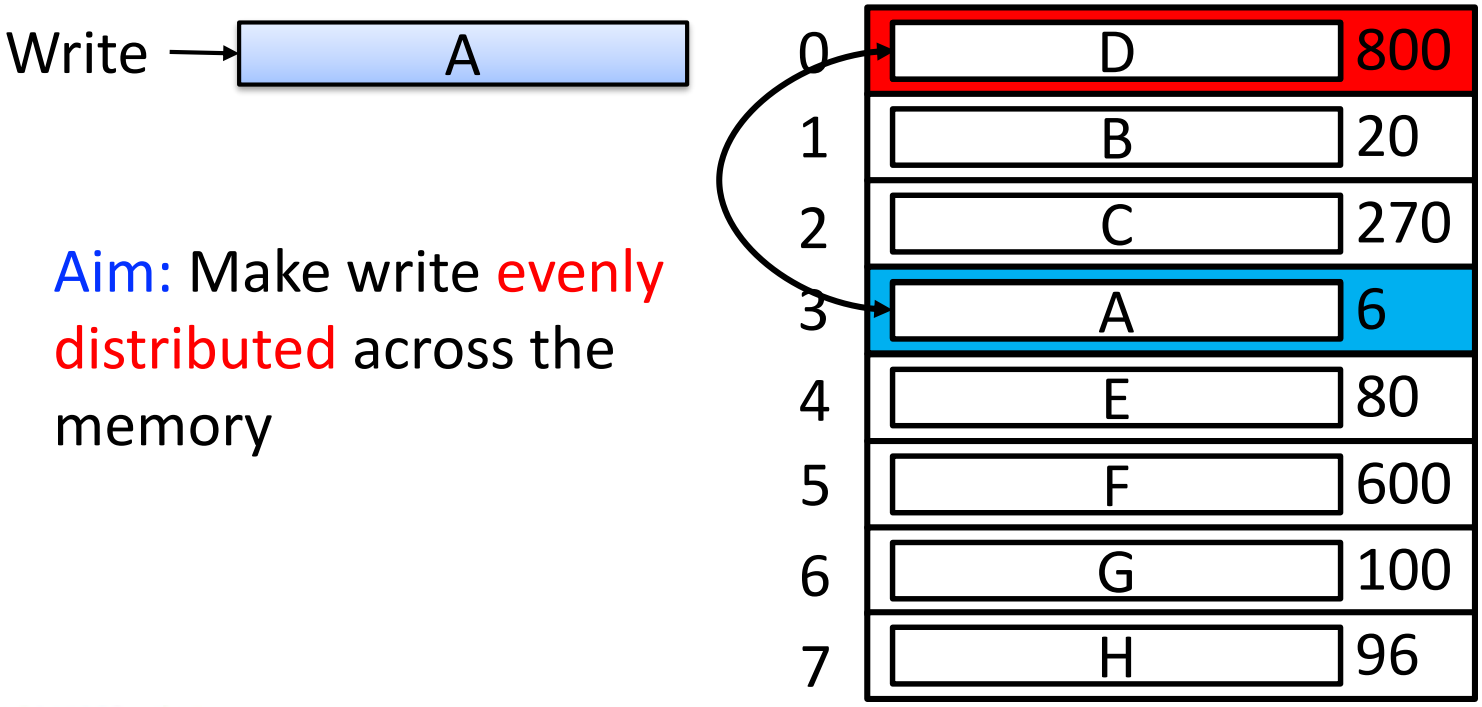
3DXpoint

- Advantages
  - High-Density: Easy to scale down under 10nm
  - Non-volatile
  - In-place update
  - Low leakage power
- Major Drawback:
  - Lifetime endurance problem
  - PCM: 10^7~10^8 writes per cell
  - In practice, lifetime around 20x shorter without wear-leveling

RICE

Wear-leveling (WL)

- A technique for prolonging the service life of some kinds of erasable computer storage media
- Block migration across the memory with certain rules
  - Move high usage blocks to low usage frames



SSD WL vs. NVRAM WL

- Solid State Disk (SSD)
  - Written out-of-place
  - Granularity:
    - Read/write: page
    - Erase: block
  - Requires garbage collection
- NVRAM
  - In-place writing
  - Granularity:
    - Read/write: byte
    - No erase
  - No garbage collection

- NVRAM has more freedom and can do better
  - No complex design for garbage collection
  - Fine-grained wear-leveling
  - Allows both algebraic and full-associative logical to physical mappings

RICE

Outline

- Background
- Previous Work
- Our Contributions
  - Hierarchical Ouroboros Wear-leveling
  - System Design
    - Architecture
    - Parameter selection
  - Experiments and Results
- Conclusion

RICE

Previous Work: NVRAM

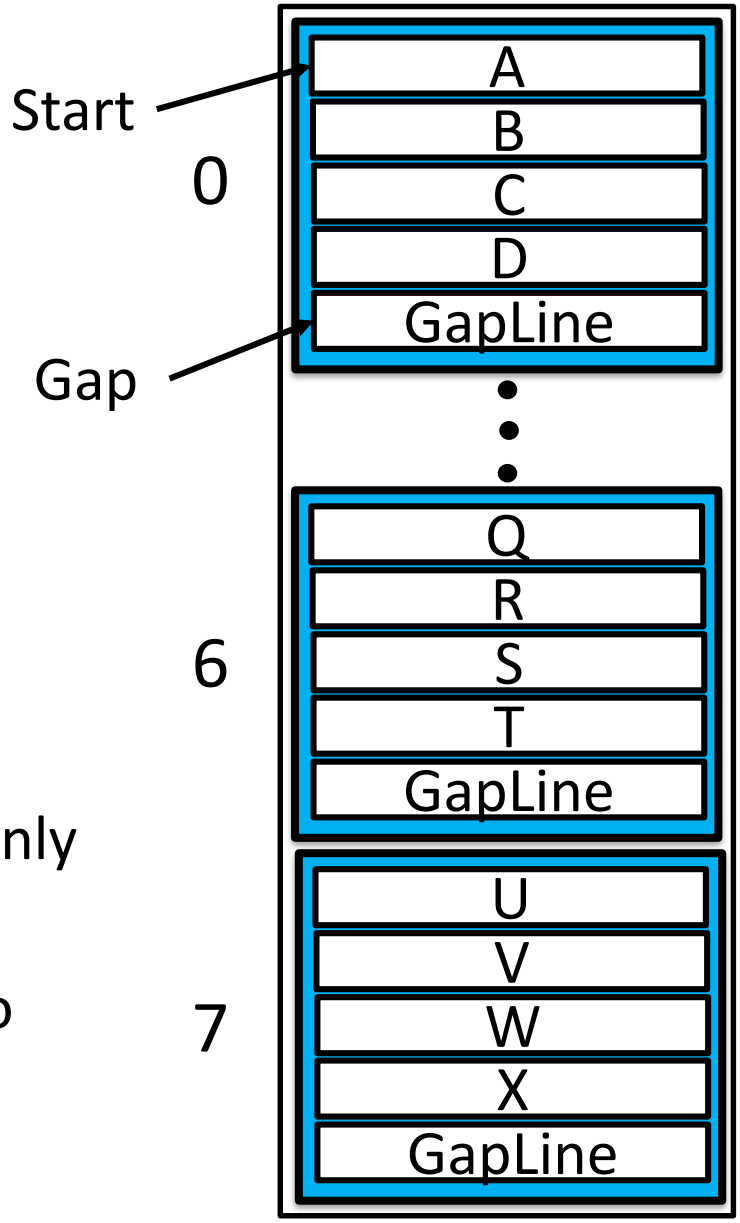
- Wear-leveling using restricted algebraic mappings
  - No address mapping table
  - Granularity: memory line (cache line)
  - Example: Start-Gap Wear-leveling [1]

- Wear-leveling using fully-associative mappings
  - Additional address mapping table needed
  - Granularity: block
  - Example: Segment Swapping [2], PCM-aware swap [3]

[1] Qureshi etal, "Enhancing lifetime and security of PCM-based main memory with start-gap wear-leveling." MICRO, 2009.
[2] Zhou etal, "A durable and energy efficient main memory using phase change memory technology" ISCA, 2009.
[3] A. P. Ferreira, M. Zhou, S. Bock, B. Childers, R. Melhem, and D. Moss´e, "Increasing pcm main memory lifetime," in Proceedings of the conference on design, automation and test in Europe. European Design and Automation Association, 2010, pp. 914–919.

Start-Gap Method Analysis

- Advantages:
  - Distribute writes smoothly within the frame
  - Small space overhead
  - Simple algorithm
- Disadvantages:
  - Region size is limited since only 1 line is relocated at a time
  - May not use all the region to distribute the writes



RICE

Previous Work: NVRAM

- Wear-leveling using restricted algebraic mappings
  - No address mapping table
  - Granularity: memory line (cache line)
  - Example: Start-Gap Wear-leveling [1]

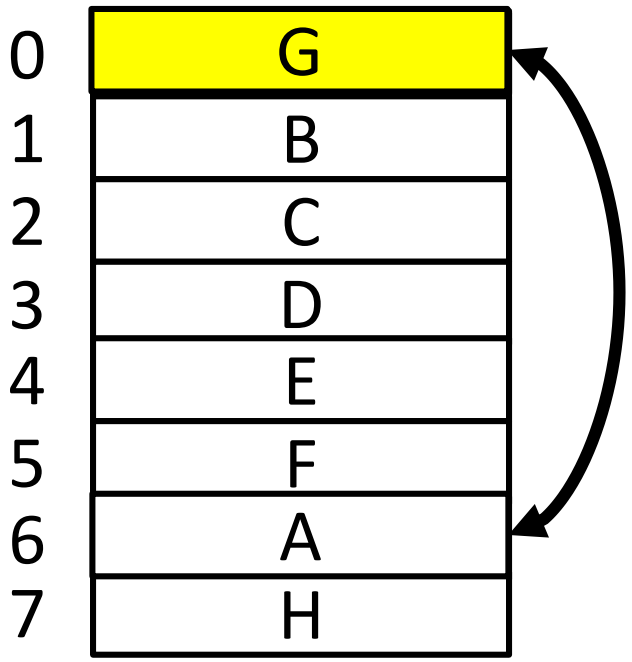
- Wear-leveling using fully-associative mappings
  - Additional address mapping table needed
  - Granularity: block
  - Example: Segment Swapping [2], PCM-aware swap [3]

[1] Qureshi etal, "Enhancing lifetime and security of PCM-based main memory with start-gap wear-leveling." MICRO, 2009.
[2] Zhou etal, "A durable and energy efficient main memory using phase change memory technology" ISCA, 2009.
[3] A. P. Ferreira, M. Zhou, S. Bock, B. Childers, R. Melhem, and D. Moss´e, "Increasing pcm main memory lifetime," in Proceedings of the conference on design, automation and test in Europe. European Design and Automation Association, 2010, pp. 914–919.

Segment Swap vs. PCM-aware Swap

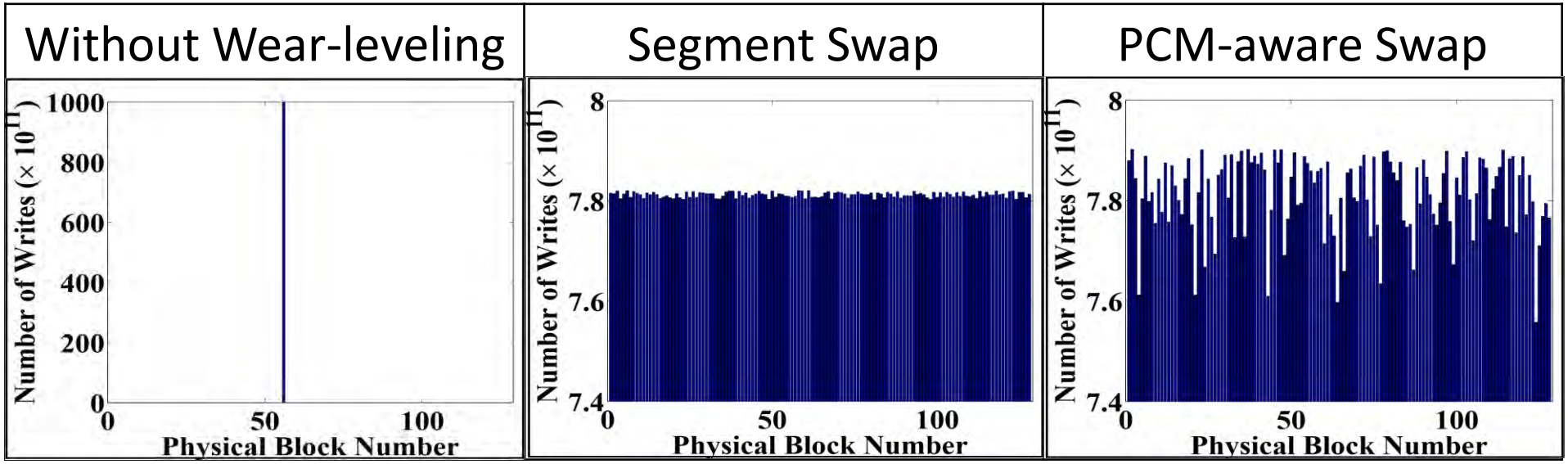
- Segment Swap:
  - Periodically swap content in highest-usage frame with content in lowest-usage frame
- PCM-aware Swap:
  - Periodically swap content in highest-usage frame with content in random frame

- Advantages:
  - Can involve all space into wear-leveling
  - Can easily be implemented



RICE

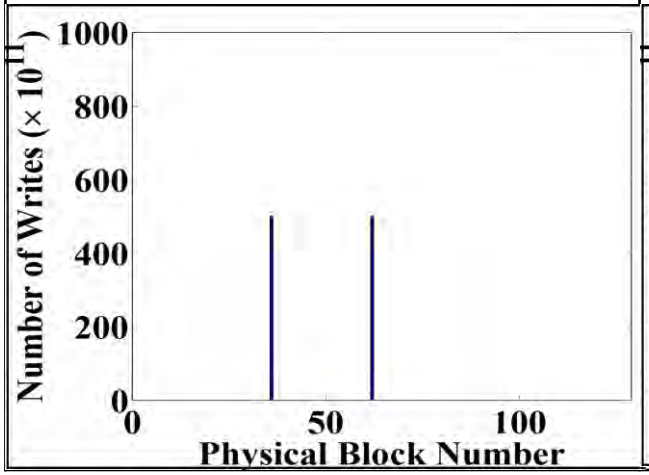
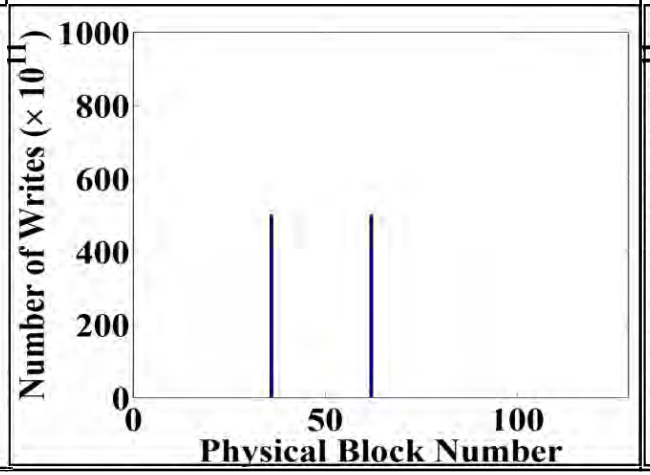
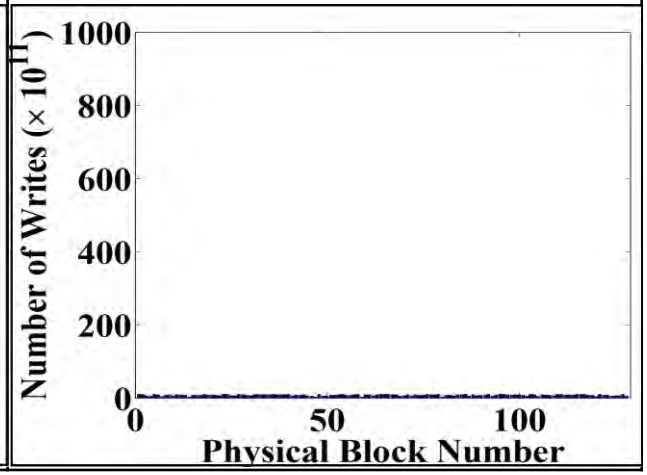
Analysis of 2 Swap Methods: A* Pattern



- A* Pattern: Write to the same logical block A continuously
- Deterministic swap is better than randomized swap under correct conditions

RICE

Analysis of 2 Swap Methods: AB* Pattern

Without Wear-leveling	Segment Swap	PCM-aware Swap
		

- AB* Pattern: Alternate writes to two logical blocks A and B (catastrophic pattern for Segment Swap)
- Randomized swap is better than deterministic swap in bad cases

RICE

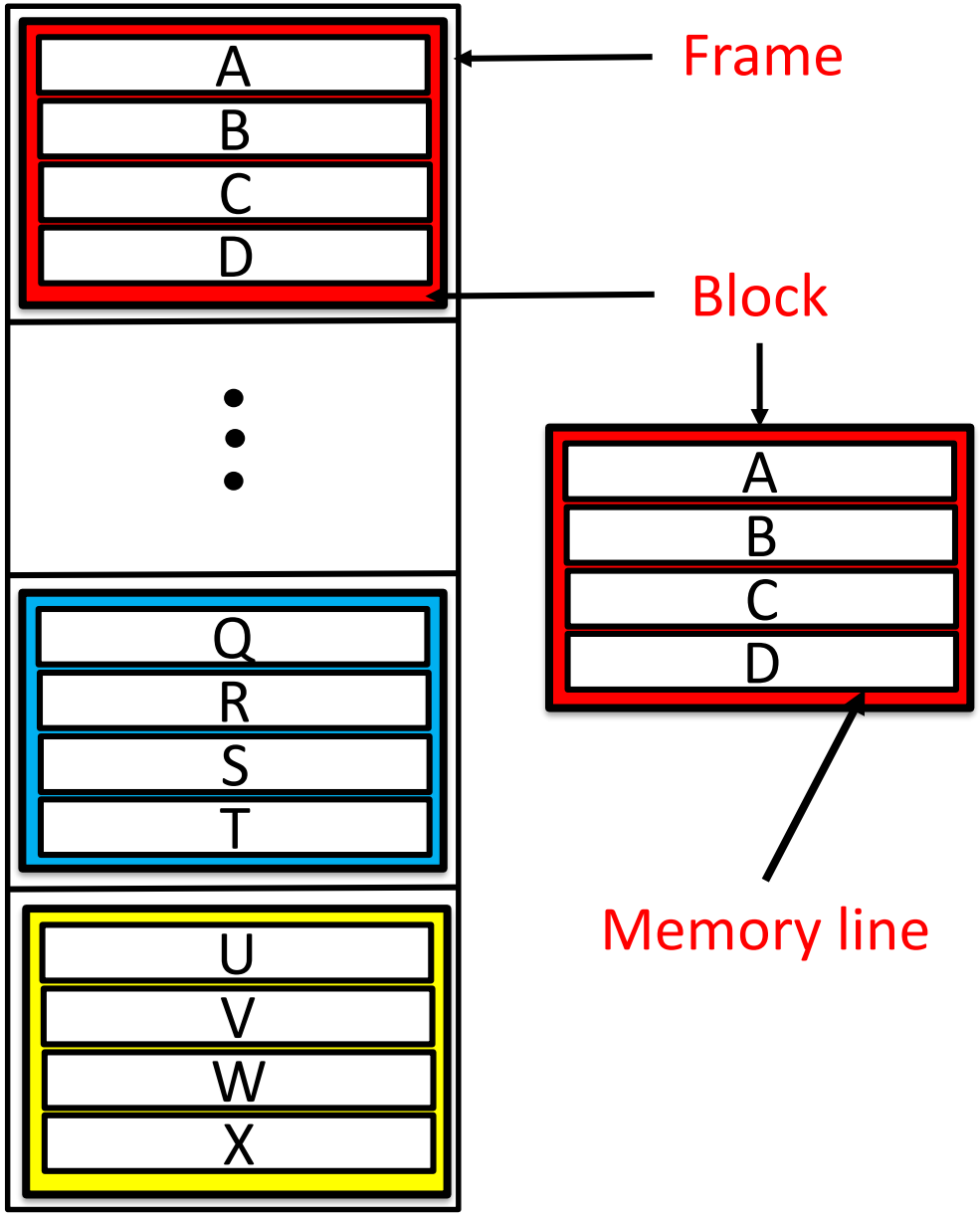
Outline

- Background
- Previous Work
- Our Contributions
  - Hierarchical Ouroboros Wear-leveling
  - System Design
    - Architecture
    - Parameter selection
  - Experiments and Results
- Conclusion

RICE

NVRAM Model

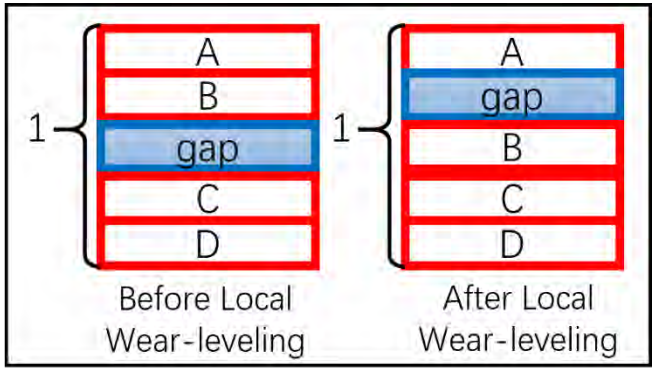
- Memory partitioned into frames
- Each frame holds a block
- A block holds a set of memory lines
- Block assumed to have consecutive address range



Hierarchical Ouroboros Wear-leveling

- Aim:
  - Guarantee write distribution as smooth as possible

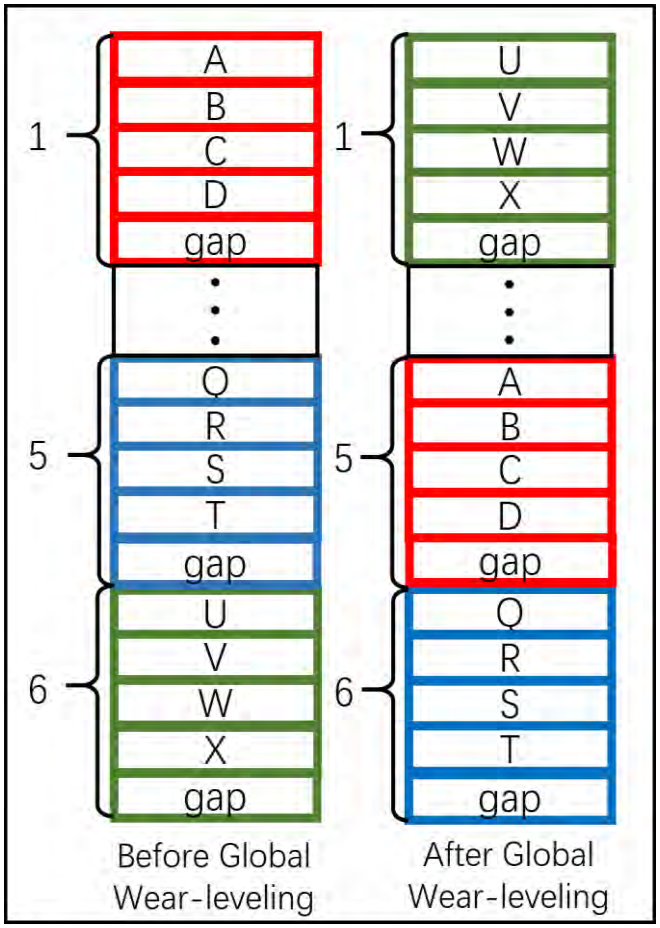
- Level 1: Local WL within frames
  - Start-gap like rule
  - Smooth distribution of writes within a frame
  - Granularity: Memory line
  - Aim: Make expensive large block Global WL less frequent



RICE

Hierarchical Ouroboros Wear-leveling

- Level 2: Global WL across frames
  - Exploit demand prediction to direct global wear-leveling
  - Use randomization in block migration to avoid worst-case behavior
  - Smooth distribution of writes across frames
  - Granularity: Frame
  - Aim: Involve all memory space into wear-leveling



Global Wear-Leveling Framework

Demand-based Ouroboros Migration

- Inputs
  1. Usage counter of each physical frame (U)
  2. Prediction of the number of future writes to each logical block (P)
     - Repetitive workloads
     - Program Analysis (embedded applications)
     - Use recent activity (demand) as predictor

RICE

Global Wear-Leveling Framework

1. Collect statistics:
   - Estimate future demand of each block to form a vector P
   - Collect current usage for each frame to form a vector U
2. Generate raw block migration mapping
   - Aim: Map the i^{th} hottest (highest demand) block to the i^{th} coldest (lowest usage) frame

RICE

Raw Block Migration

Initialization:

Physical Frame(Usage U)

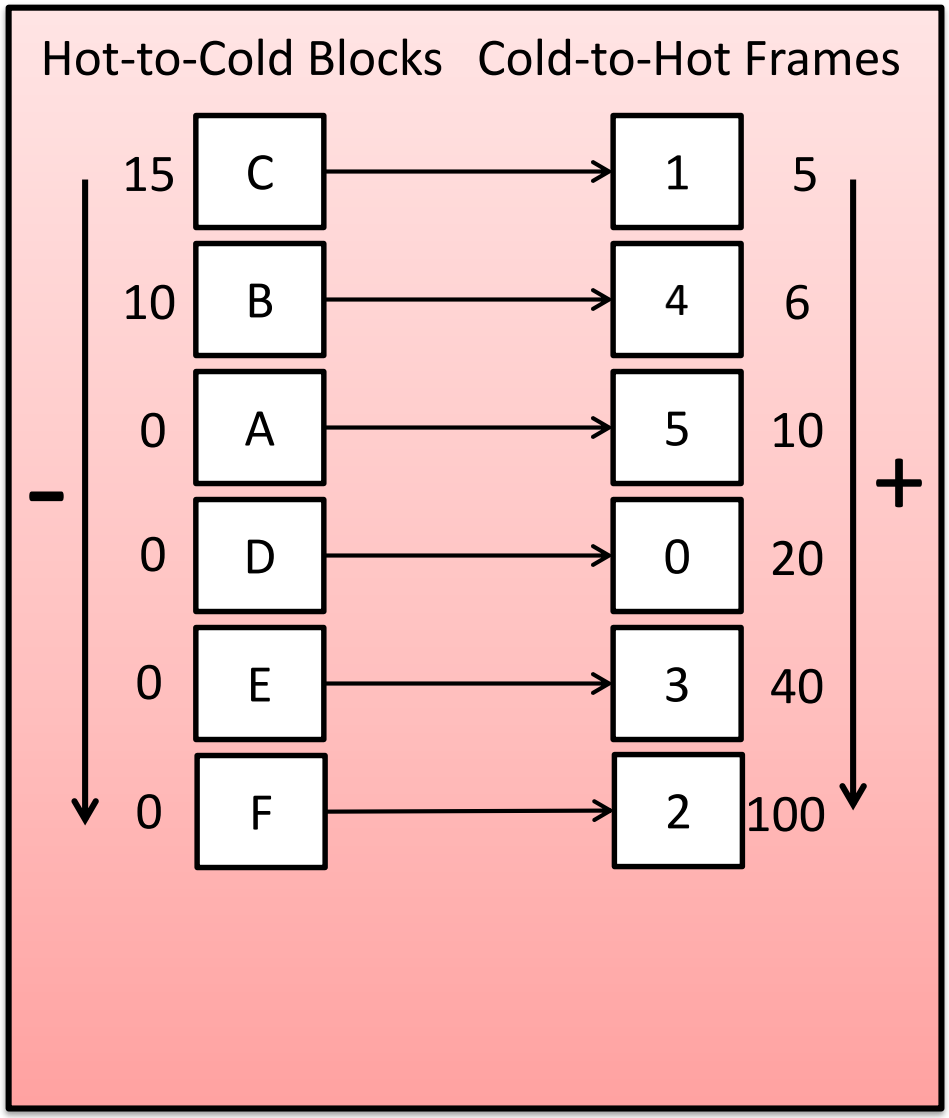
0	1	2	3	4	5
20	5	100	40	6	10

Logical Block (Demand P)

0	1	2	3	4	5
A	B	C	D	E	F
0	10	15	0	0	0

0	1	2	3	4	5
D	C	F	E	B	A

Final Block Order



Global Wear-Leveling Framework

1. Collect statistics:
   - Estimate future demand of each block to form a vector A
   - Collect current usage for each frame to form a vector U
2. Generate raw block migration mapping
   - Aim: Map the i^{th} hottest (highest demand) block to the i^{th} coldest (lowest usage) frame
3. Classification step:
   - Identify a hot pool with up to K hottest blocks that meet a minimum demand threshold
4. Pruning Step:
   - Move only blocks in the hot pool to deterministic frames

RICE

Block Migration with Pruning Method

Initialization:

Physical Frame(Usage U)

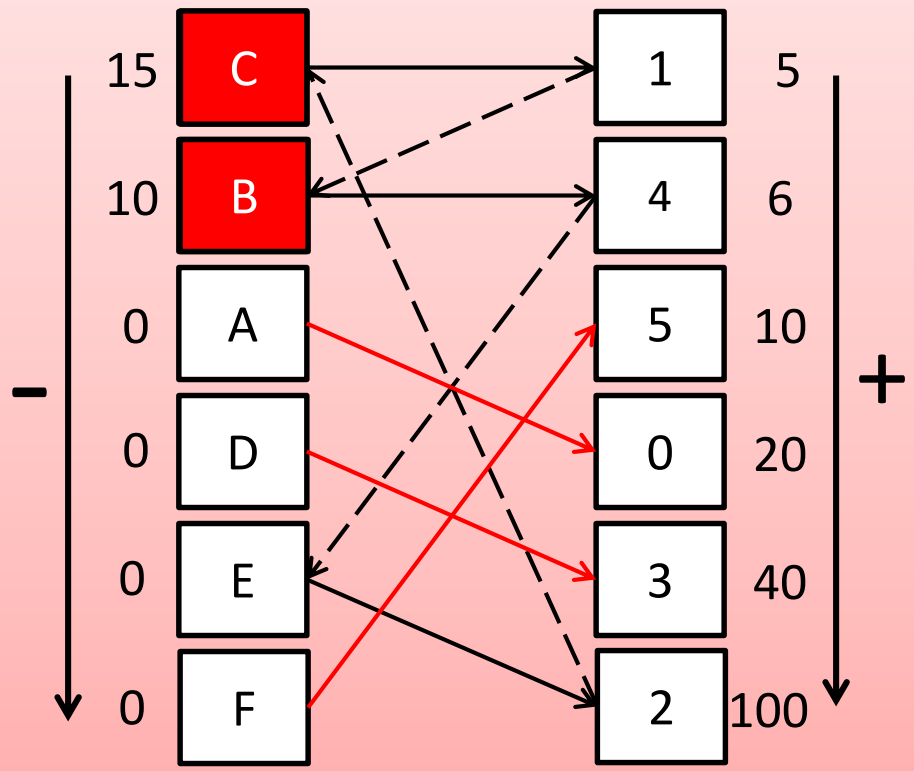
0	1	2	3	4	5
20	5	100	40	6	10

Logical Block (Demand D)

0	1	2	3	4	5
A	B	C	D	E	F
0	10	15	0	0	0

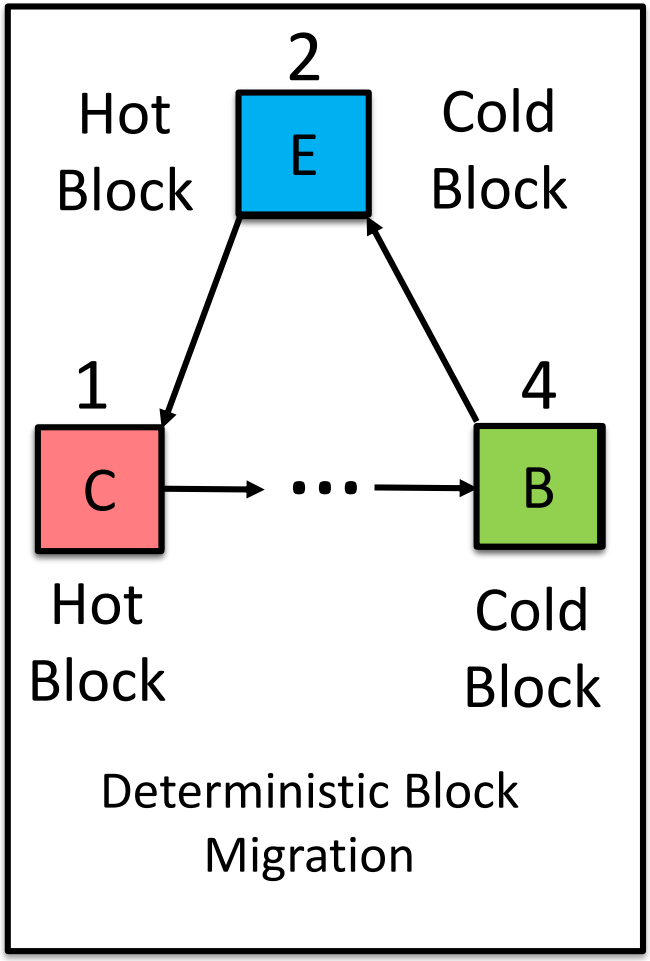
0	1	2	3	4	5
A	C	E	D	B	F

Final Block Order:

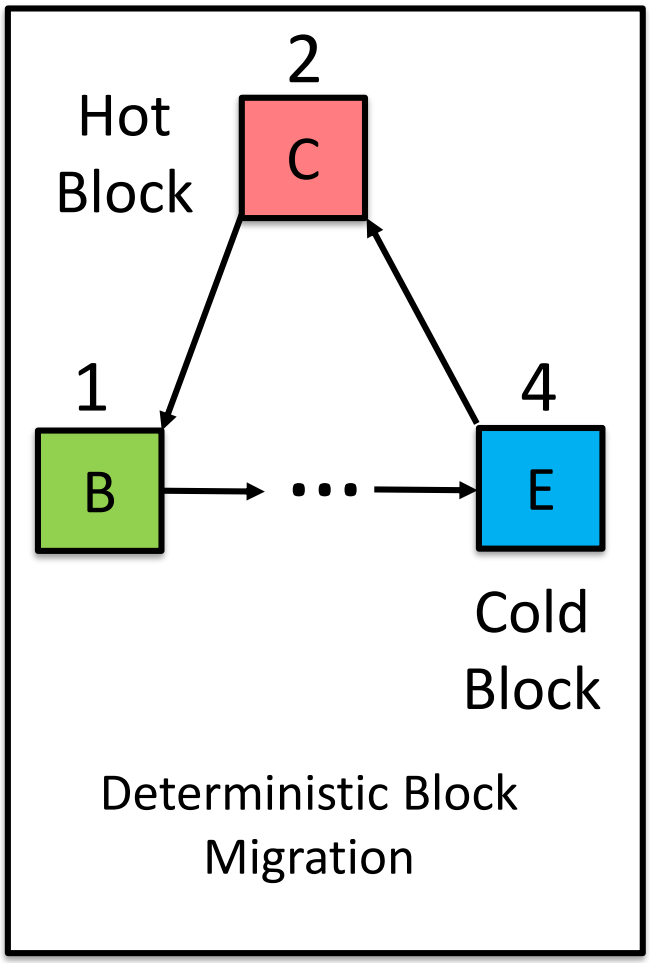


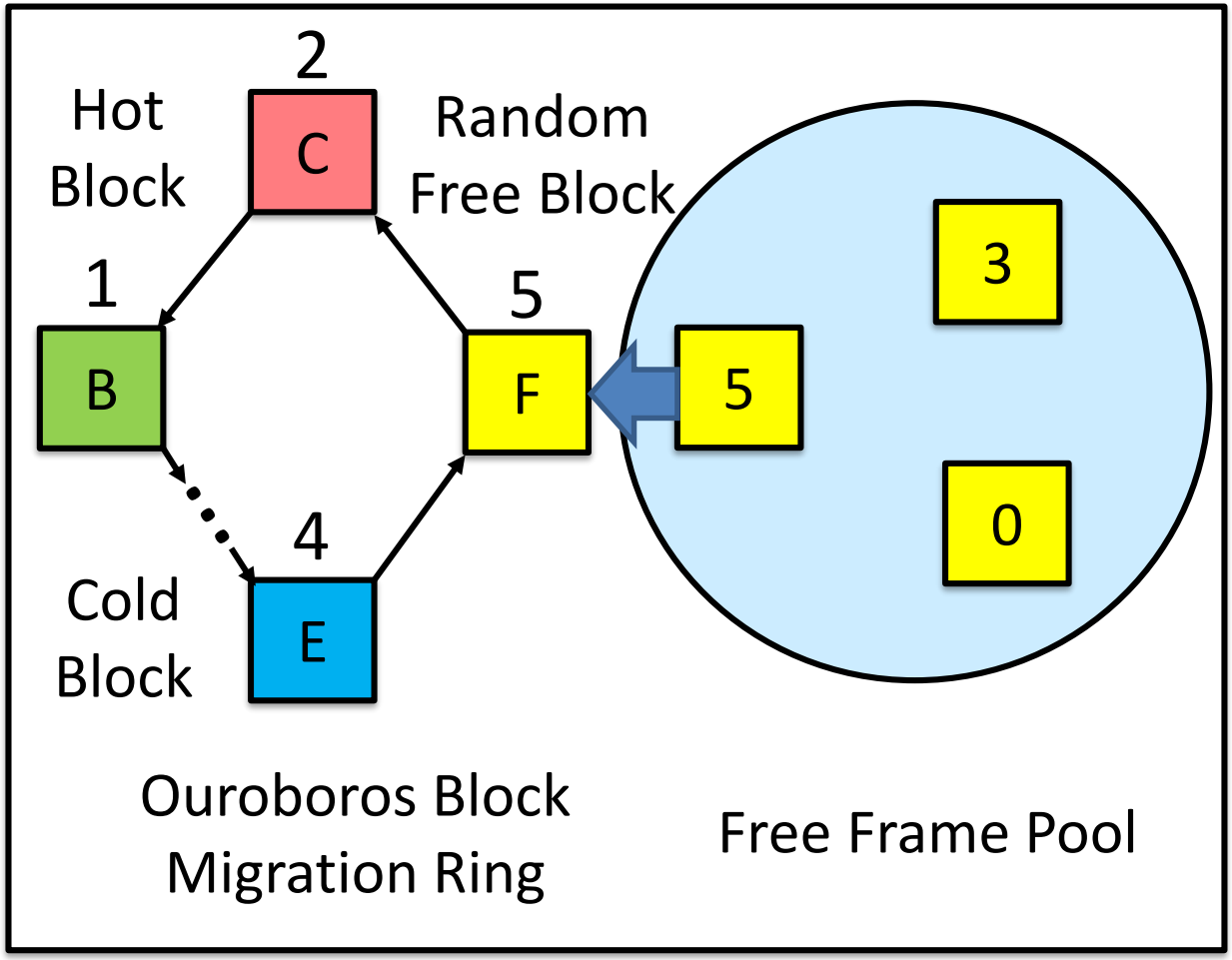
$|H| = |C| = 2$

Deterministic Block Migration Ring

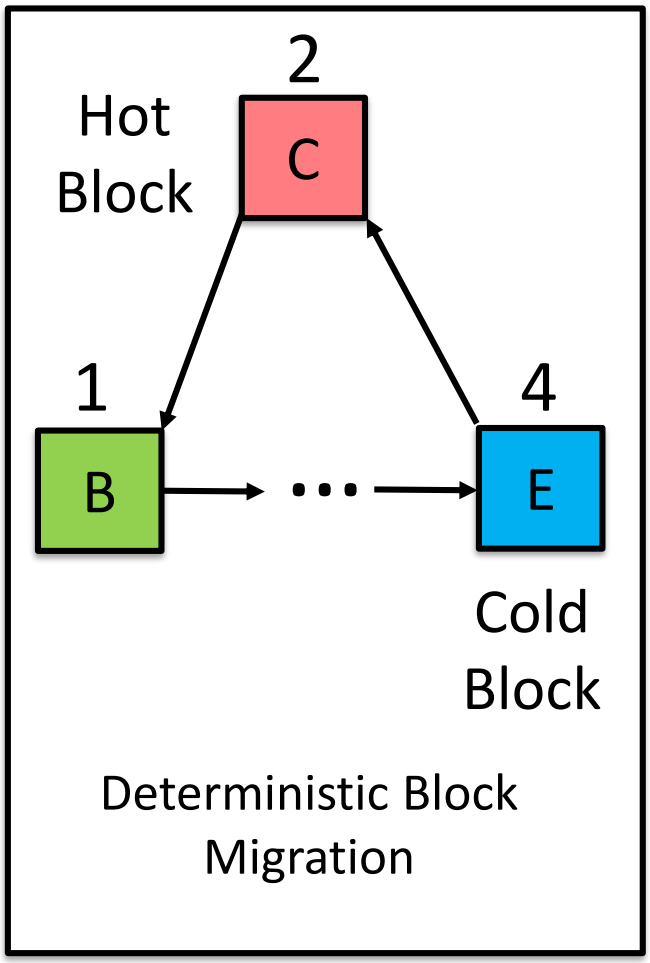


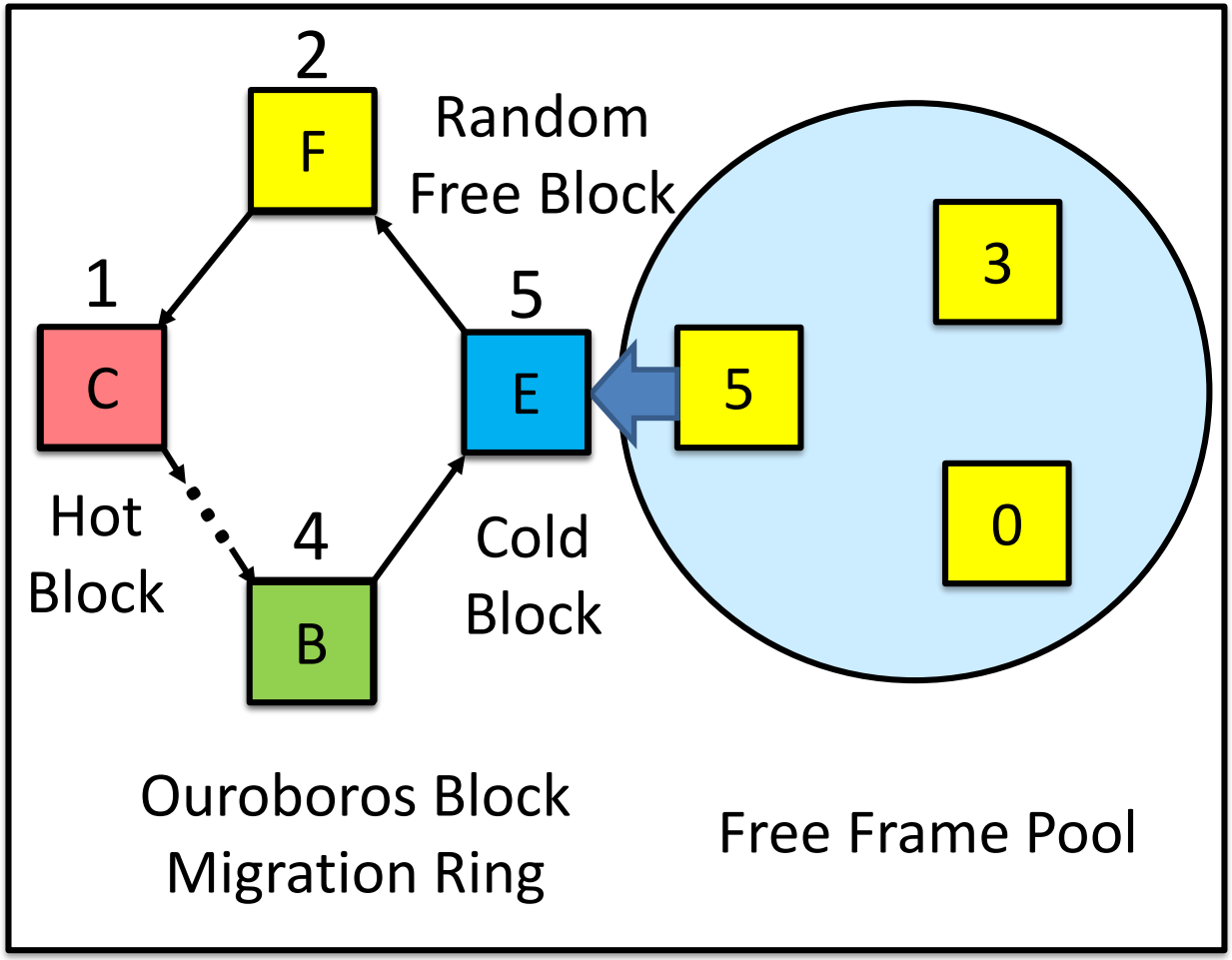
Ouroboros Block Migration Ring





Ouroboros Block Migration Ring





Global Wear-Leveling Framework

1. Collect statistics:
   - Estimate future demand of each block to form a vector A
   - Collect current usage for each frame to form a vector U
2. Generate raw block migration mapping
   - Aim: Map the i^{th} hottest (highest demand) block to the i^{th} coldest (lowest usage) frame
3. Classification step:
   - Identify a hot pool with up to K hottest blocks that meet a minimum demand threshold
4. Pruning Step:
   - Move only blocks in the hot pool to deterministic frames
5. Randomization step:
   - Identify free frame pool with more than K free frames for randomization
6. Form Ouroboros block migration ring for block relocation

Block Migration with Randomization

Initialization:

Physical Frame(Usage U)

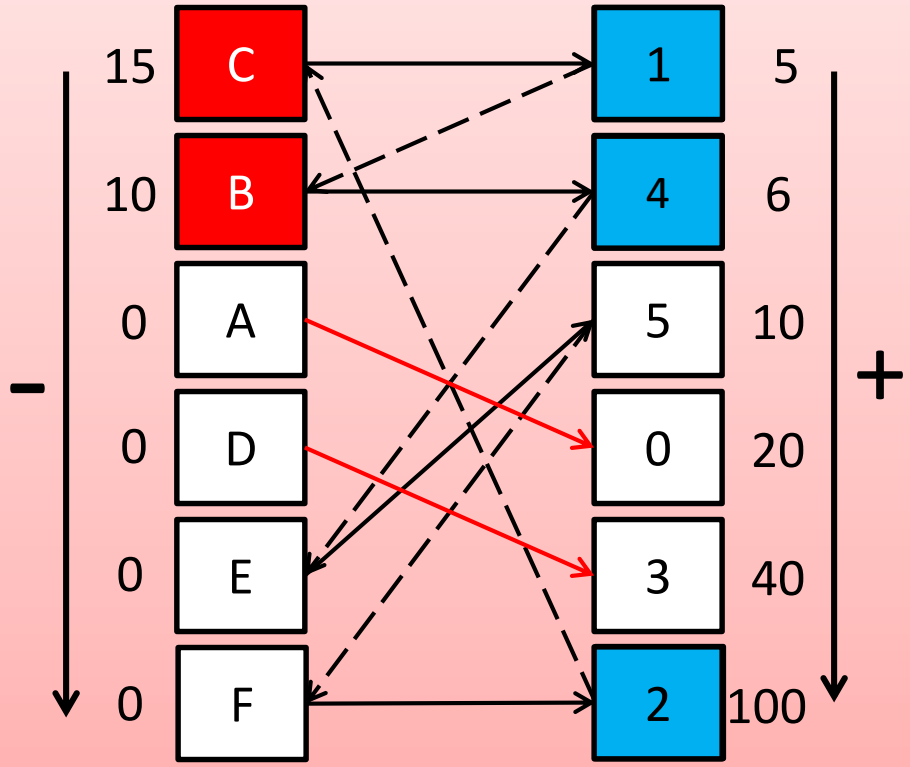
0	1	2	3	4	5
20	5	100	40	6	10

Logical Block (Demand D)

0	1	2	3	4	5
A	B	C	D	E	F
0	10	15	0	0	0

0	1	2	3	4	5
A	C	F	D	B	E

Final Block Order:



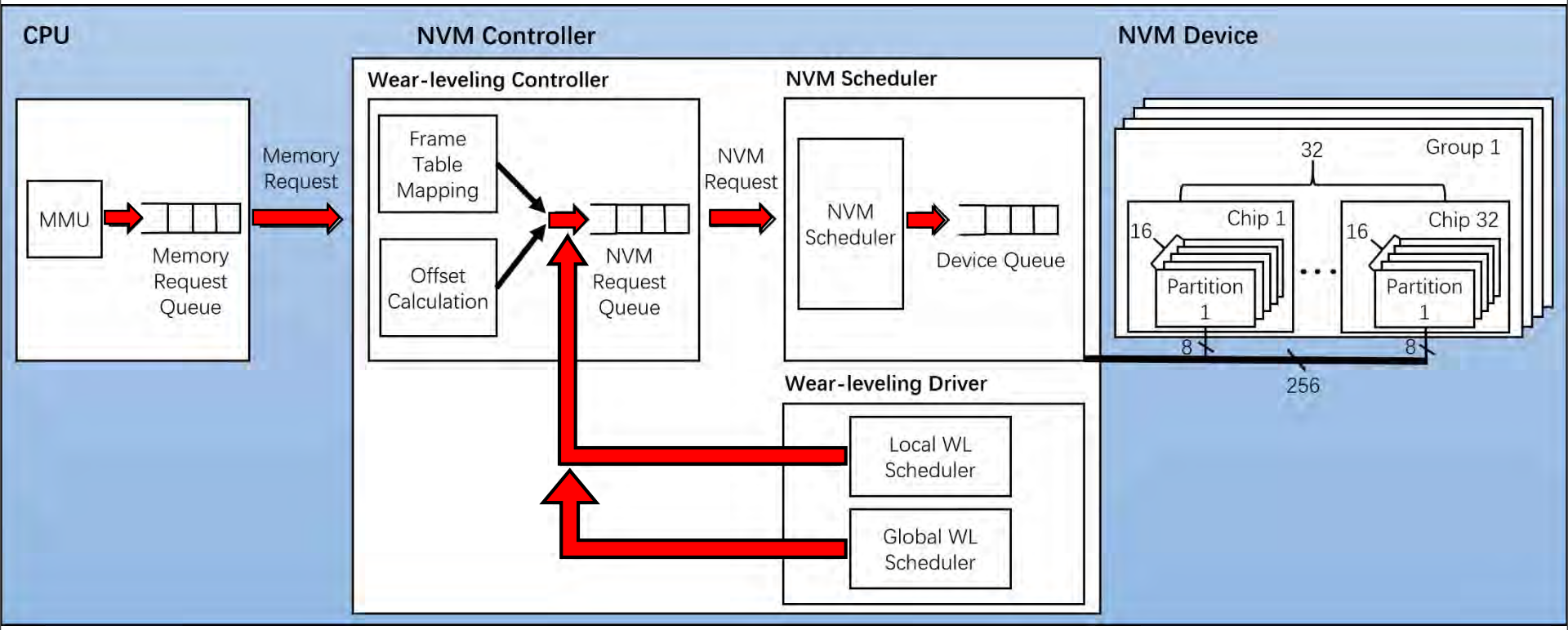
|H| = |C| =2, |F|=2

Outline

- Background
- Previous Work
- Our Contributions
  - Hierarchical Ouroboros Wear-leveling
  - System Design
    - Architecture
    - Parameter selection
  - Experiments and Results
- Conclusion

RICE

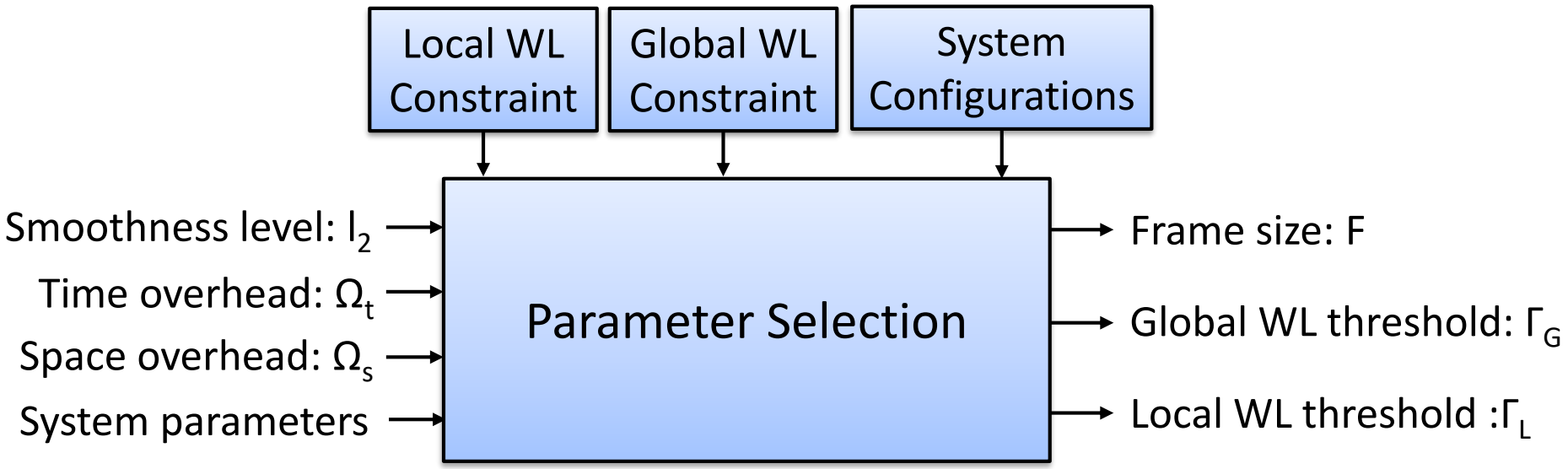
Architecture



- Each request
  - Size 16B * 32 = 512B
  - Touch same partition and offset for all 32 chips

RICE

Parameter Selection



- Example: Parameter Selection for 512GB Memory
  - Input:
    - l_2 : $7x10^{-6}$, Ω_t : 0.6%, Ω_s : 0.5%
  - Output:
    - F: 8KB, Γ_G: $1x10^8$, Γ_L: 195
  - Worst case overhead: Ω_t : 0.52%, Ω_s : 0.2%

RICE

Outline

- Background
- Previous Work
- Our Contributions
  - Hierarchical Ouroboros Wear-leveling
  - System Design
    - Architecture
    - Parameter selection
  - Experiments and Results
- Conclusion

RICE

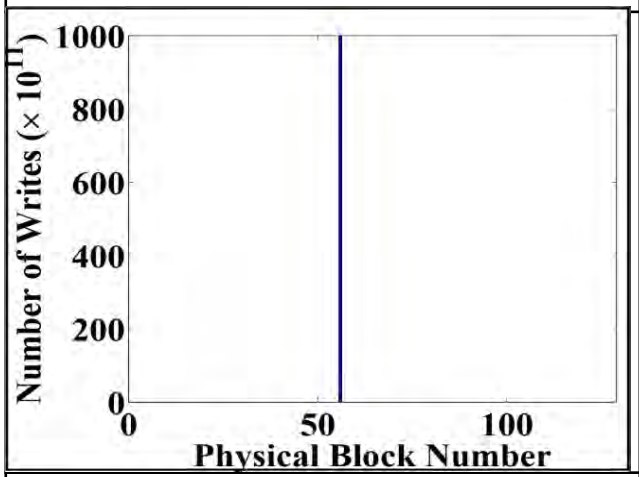
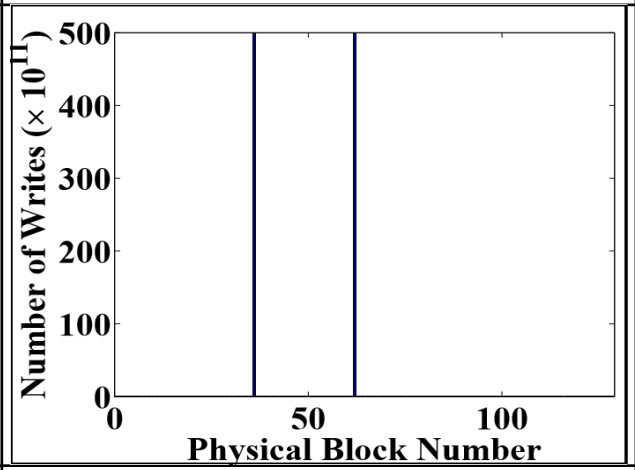
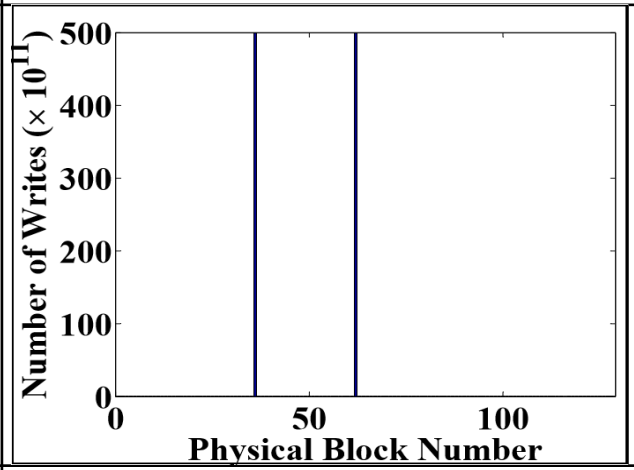
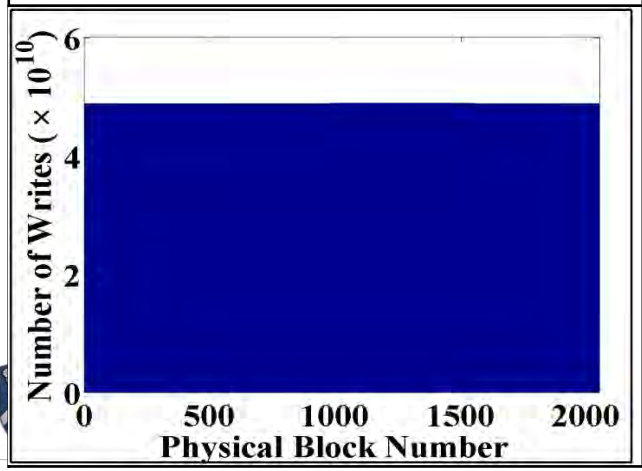
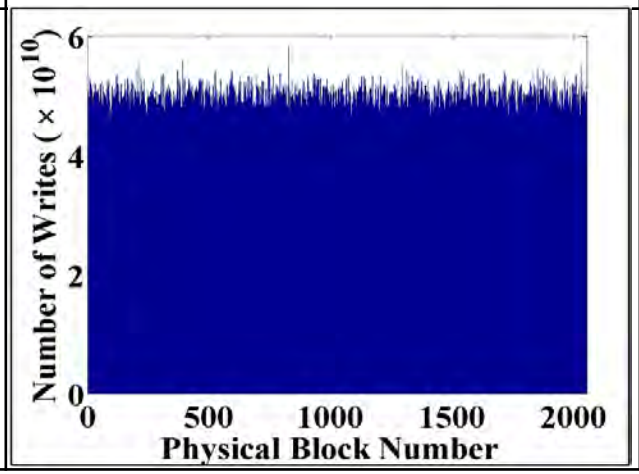
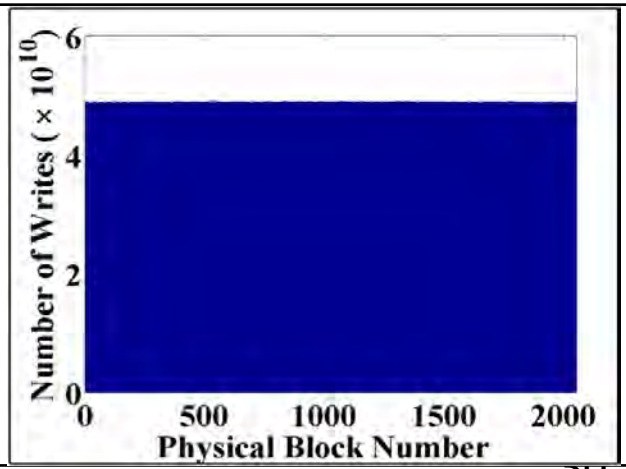
Experiments

- Smoothness value:
  - L_∞ smoothness: $L_\infty = \max_i |u_i - \hat{u}_i|$
  - L_2 smoothness: $L_2 = \sqrt{\frac{\sum_{i=1}^{N_B} (\frac{u_i - \hat{u}_i}{W})^2}{N_B}}$
- Usage Distribution
- Experiments
  - Micro Benchmarks:
    - A* pattern, AB* pattern, AB*50% pattern
    - Total writes: 10^{14}
  - Storage Benchmarks:
    - MSR Cambridge pattern, FIU IODedup pattern
    - Total writes per chip: 2.83 x 10^{12}
    - Write rate per chip: 500MB/s x 32

Note: u_i is the real usage distribution, $\hat{u}_i$ is the ideal usage distribution, W is the total number of writes

Parameters	Micro	Storage
NVM size (M)	512 MB	512 GB
Frame size (F)	8 KB	8 KB
Line size (L)	16 bytes	16 bytes
Stripe size (C)	32 chips	32 chips
Local Threshold (Γ_L)	195	195
Global Threshold (Γ_G)	1×10^7	1×10^8

Micro Experiments Results

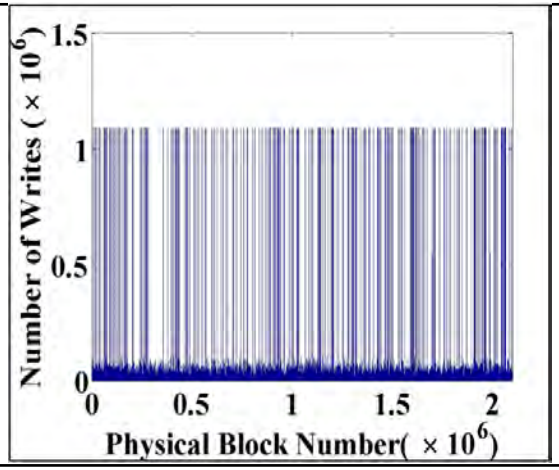
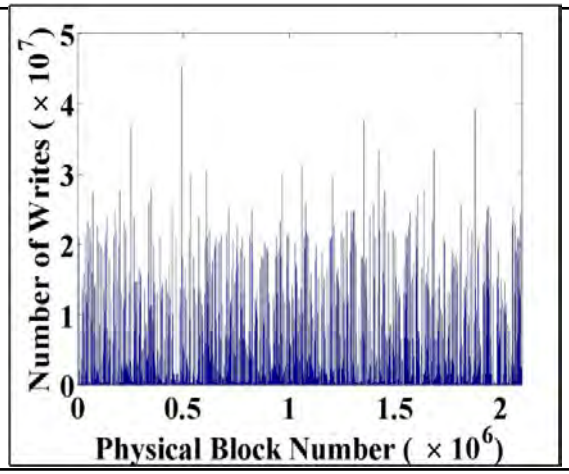
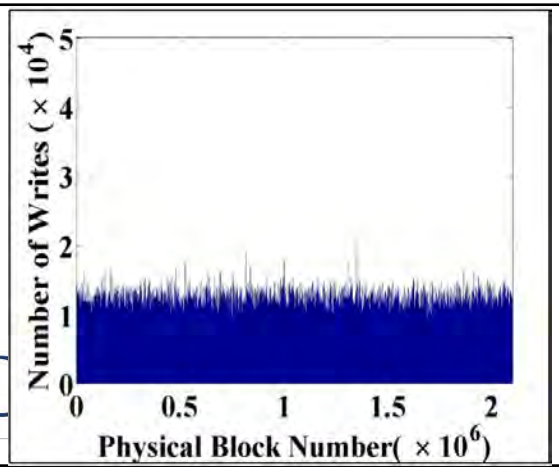
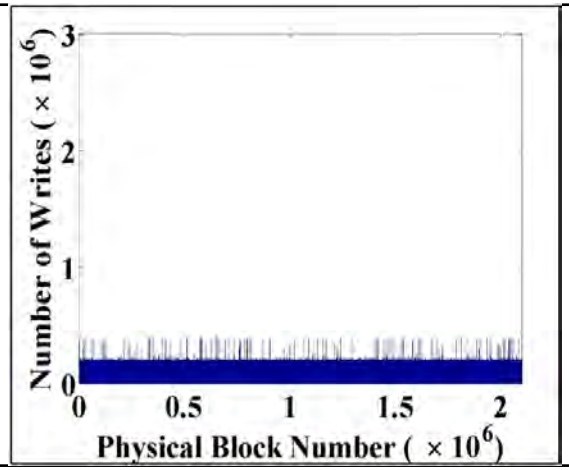
A* Pattern	AB* Pattern	(AB)*50% Pattern
Without Wear-leveling	Without Wear-leveling	Without Wear-leveling
		
After Wear-leveling	After Wear-leveling	After Wear-leveling
		

Summary of Ourobros WL

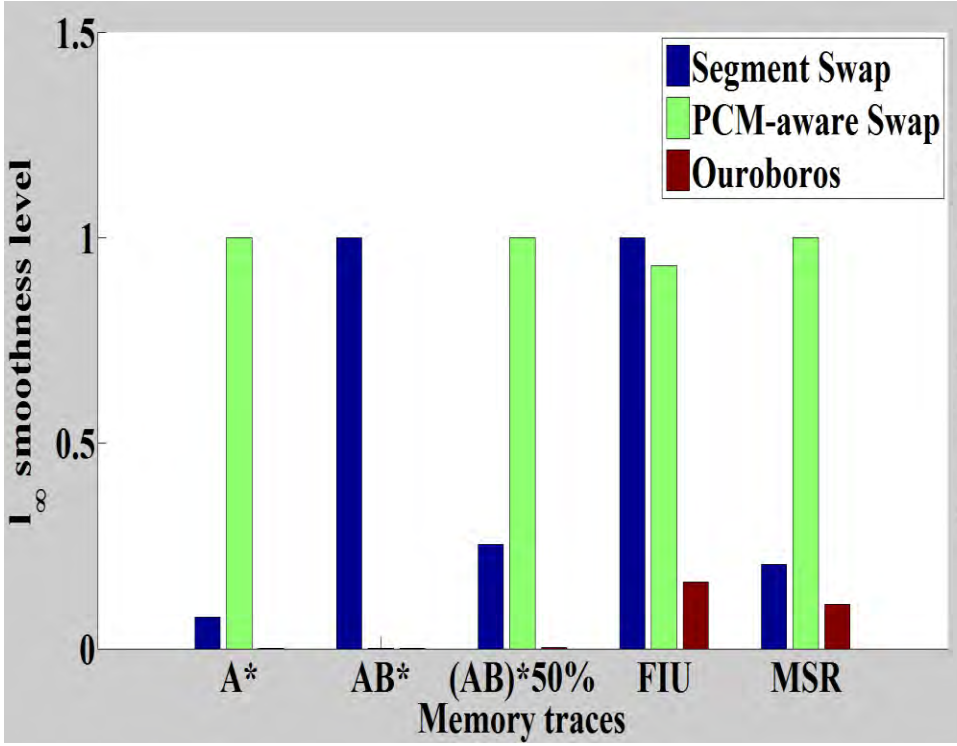
- Correct prediction: Achieve the best possible smoothness behavior
- Wrong prediction: No worse than the distribution obtained by a random write pattern
- Partial correct prediction: Fully take advantages of correct prediction to make usage distribution smooth

RICE

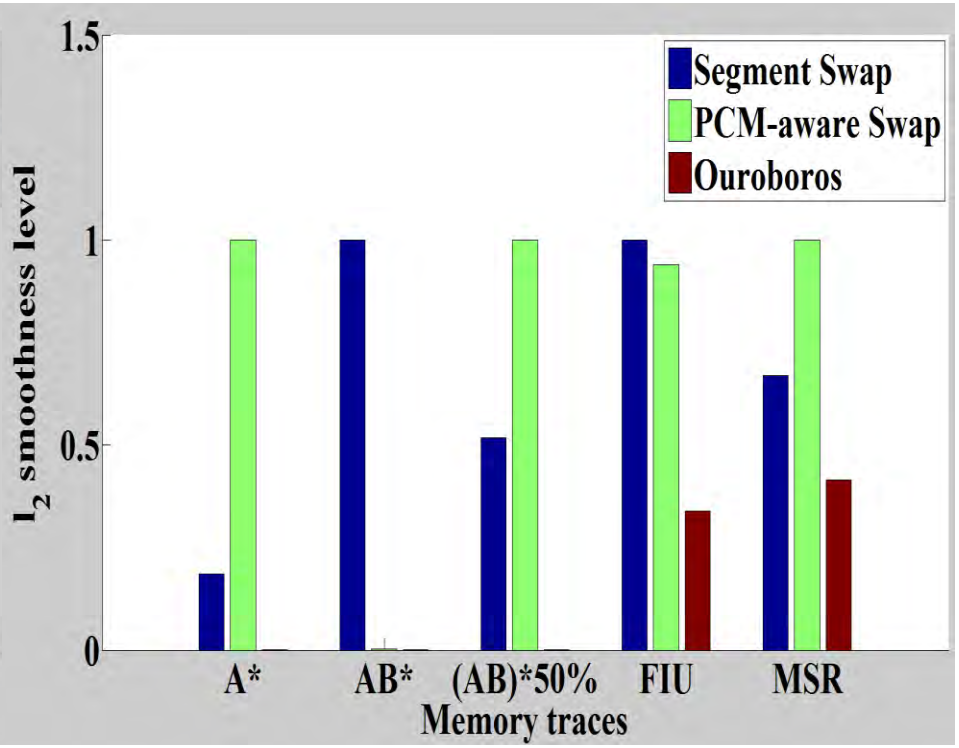
Storage Experiments Results

MSR Cambridge	FIU IODedup
Without Wear-leveling	Without Wear-leveling
	
After Wear-leveling	After Wear-leveling
	

Comparison Among Three WL Methods



l_∞ smoothness level



l_2 smoothness level

RICE

Outline

- Background
- Previous Work
- Our Contributions
  - Hierarchical Ouroboros Wear-leveling
  - System Design
    - Architecture
    - Parameter selection
  - Experiments and Results
- Conclusion

RICE

Conclusion

- Design a Hierarchical Ouroboros Wear-leveling Method
  - Memory line level Local Wear-leveling
  - Frame level Global Wear-leveling
- Devise a cyclic block migration method
  - Deterministically smooth wear out based on prediction
  - Involve randomization to break up destructive write pattern
- Show Ouroboros wear-leveling system architecture
- Provide a general way to select parameter settings
- Show the realizability and feasibility of Ouroboros wear-leveling through experiments

RICE