Technion | Electronics Computers Communications

NAND Flash Architectures Reducing Write Amplification Through Multi-Write Codes

Saher Odeh
Yuval Cassuto
Technion-EE

MSST 2014
June 6th

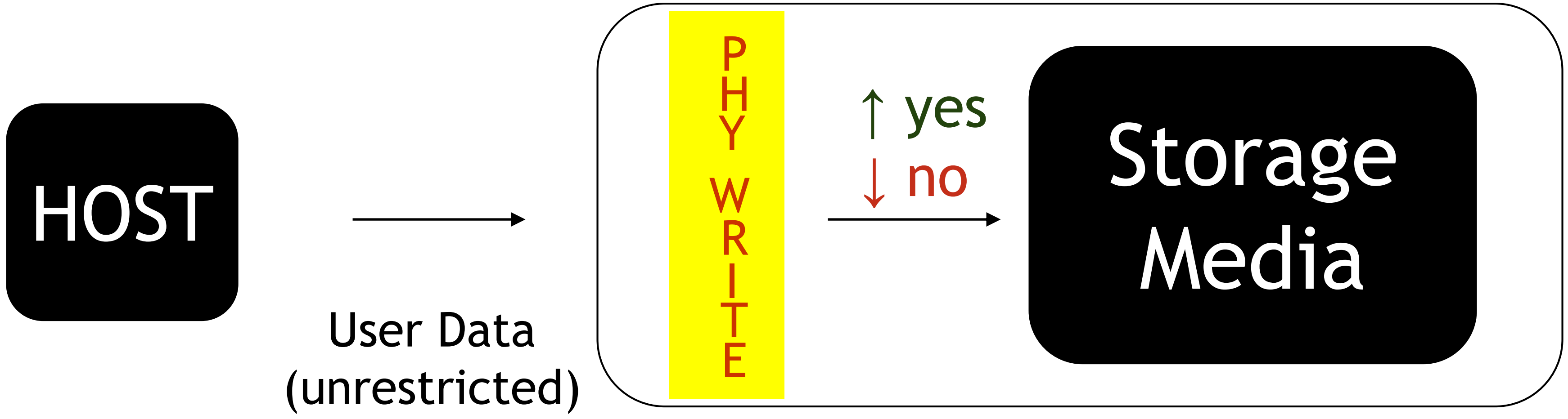
Main Objective

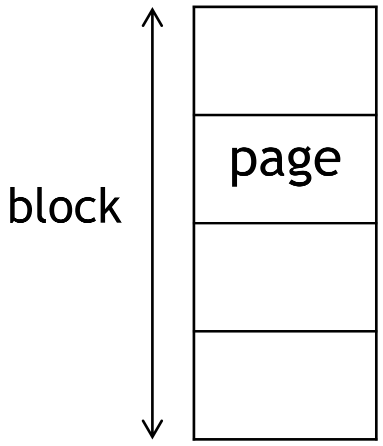
- Improve device performance - speed, power and lifetime
- Reduce device cost per bit (silicon space)
- Require minimal / no changes to NAND storage unit

Method:

- Propose architectures that cleverly use **Multi Write Codes**

Problem: No In-Place Writes





- Page - write / read unit
- Block – erasure unit

Write Amplification (WA)

- Malady: Write-Amplification
  - Adding spare blocks reduces WA (over-provisioning)
- Example:
  - Writes:

0	4	8	12	
1	5	9	13	
2	6	10	14	
3	7	11	15	

[Agarwal et. al 2010, Desnoyers 2012,…]

Write Amplification (WA)

- Malady: Write-Amplification
  - Adding spare blocks reduces WA
- Example:
  - Writes: 0,4,8,12,

~~0~~	~~4~~	~~8~~	~~12~~	0
1	5	9	13	4
2	6	10	14	8
3	7	11	15	12

[Agarwal et. al 2010, Desnoyers 2012,…]

Write Amplification (WA)

- Malady: Write-Amplification
  - Adding spare blocks reduces WA
- Example:
  - Writes: 0,4,8,12,1

2	~~4~~	~~8~~	~~12~~	0
3	5	9	13	4
1	6	10	14	8
	7	11	15	12

[Agarwal et. al 2010, Desnoyers 2012,…]

Multi-Write Codes

- Remedy: Multi-Write Codes
  - Up to $t > 1$ writes before erase!
- Example t = 2

0	4	8	12	
1	5	9	13	
2	6	10	14	
3	7	11	15	

Multi-Write Codes

- Remedy: Multi-Write Codes
  - Up to **t** writes before erase!
- Example t = 2
  - Writes: 0,1,2,4,5,6,8,9,10,12

0	4	8	12	
1	5	9	13	
2	6	10	14	
3	7	11	15	

~~4~~	Invalid
5	Written twice

Multi-Write Codes

- Remedy: Multi-Write Codes
  - Up to **t** writes before erase!
- Example t = 2
  - Writes: 0,1,2,4,5,6,8,9,10,12,0,4,8,12

~~0~~	~~4~~	~~8~~	~~12~~	0
1	5	9	13	4
2	6	10	14	8
3	7	11	15	12

- ~~4~~ — Invalid
- 5 — Written twice

Multi-Write Codes

- Remedy: Multi-Write Codes
  - Up to **t** writes before erase!
- Example t = 2
  - Writes: 0,1,2,4,5,6,8,9,10,12,0,4,8,12,1
  - # Writes : 4 → 14

2	~~4~~	~~8~~	~~12~~	0
3	5	9	13	4
1	6	10	14	8
	7	11	15	12

~~4~~	Invalid
5	Written twice

The Tradeoff of Multi-Write Codes

- More in-place writes → higher code redundancy
- Code redundancy shrinks spare available for WA reduction

- Research question: can Multi-Write codes reduce overall WA?

t=2 Multi-Write Codes

- Q: How big is the physical page for t=2 writes? (expansion-factor)
  - A: capacity-achieving WOM codes:

$$|\text{physical page}| = \frac{2\log_2 q}{\log_2\binom{q+1}{2}} |\text{logical page}|$$

q=2	SLC
q=4	MLC
q=8	TLC

- For t=2,q=2; expansion of data: ~1.26
- For t=2,q=8; expansion of data: ~1.16

Higher q → lower expansion

[Luojie, Kurkoski, Yaakobi, 2012]

Prior Work

- Multi-write + compression

[Jagmohan, Franceschini, Lastras, 2010]

- Analysis of WA w/ multi-write codes

[Luojie, Kurkoski, Yaakobi, 2012]

Known Results

[Luojie, Kurkoski, Yaakobi, 2012]

- Multi Write codes reduce WA when
  1. Spare > 75% for SLC (q=2)
  2. Spare > 55% for MLC (q=4)
  3. Spare > 40% for TLC (q=8)

Cost likely too high for SSD deployment

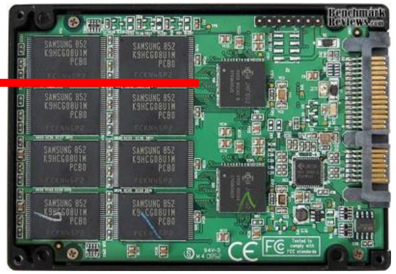
Partial Re-Write

- A natural idea:

t=2

t=1

no re-write, no overhead



- <u>The challenge:</u>

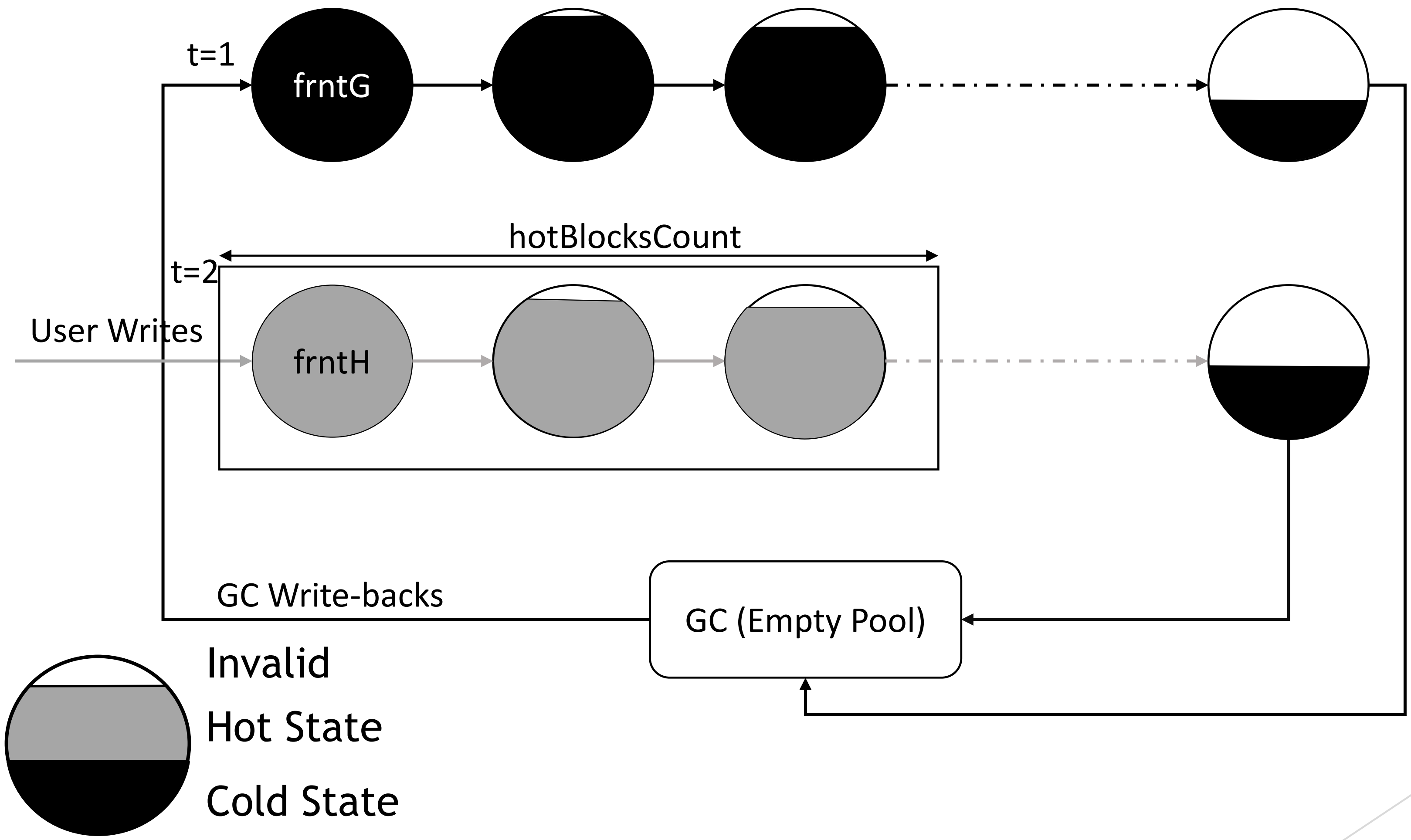
Which writes should get re-write capabilities?

<u>The Answer</u>: the writes that will be rewritten before erasure.

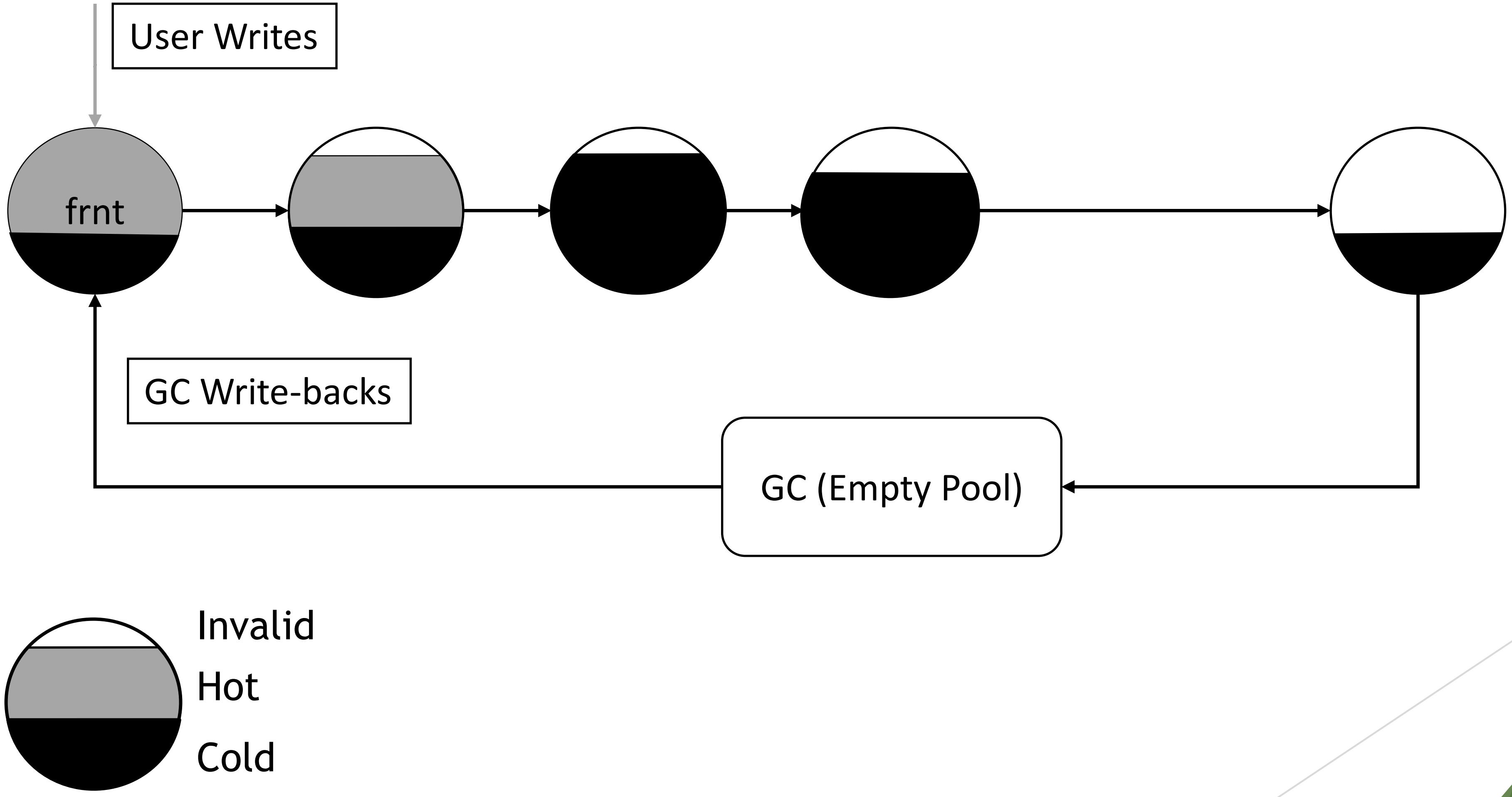
Re-Write Differentiation

- The Policy:
  - Perform t=2 coding for all incoming user-writes
  - Write without coding (t=1) all garbage-collection writes
- Observation:
  - With high likelihood, user writes will be rewritten soon (temporal-locality)

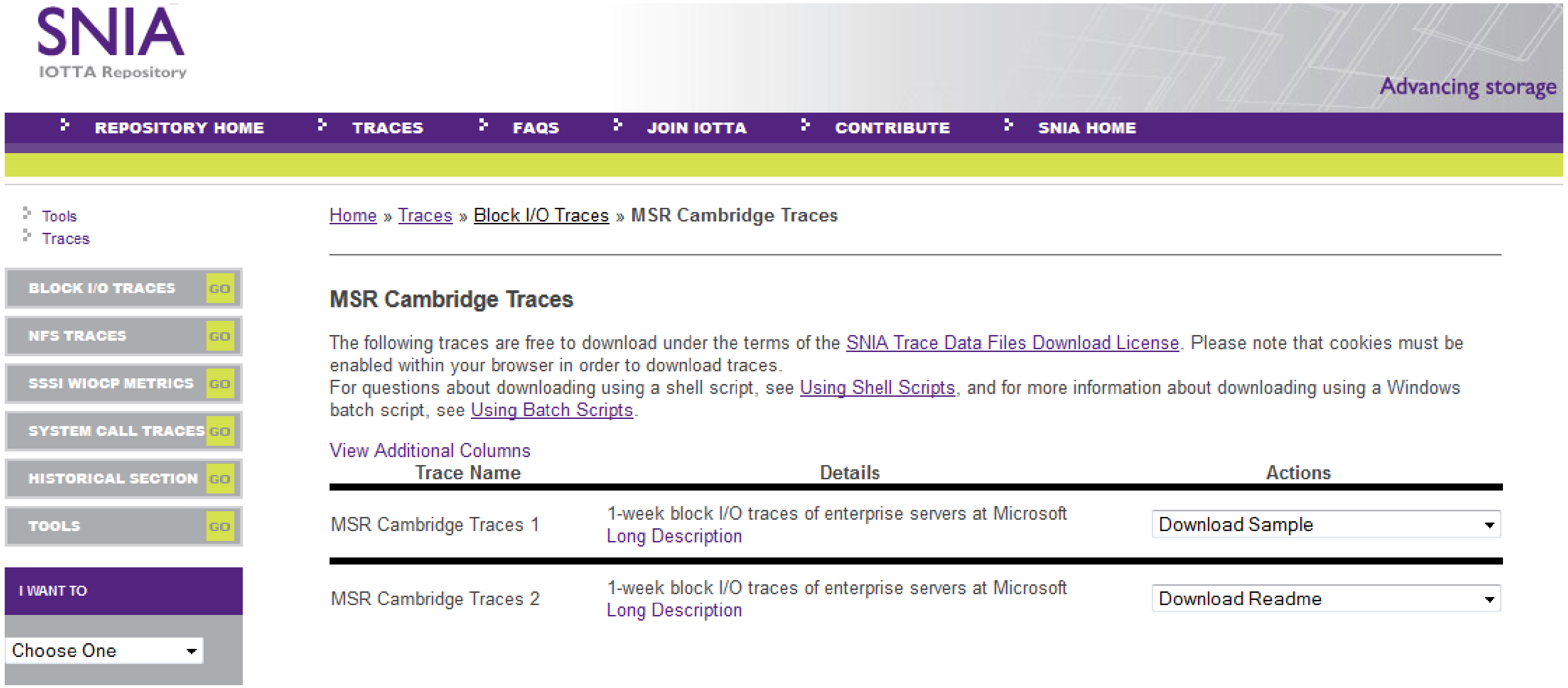
Double-Fronted Architecture



Selective (Single-Fronted) Architecture

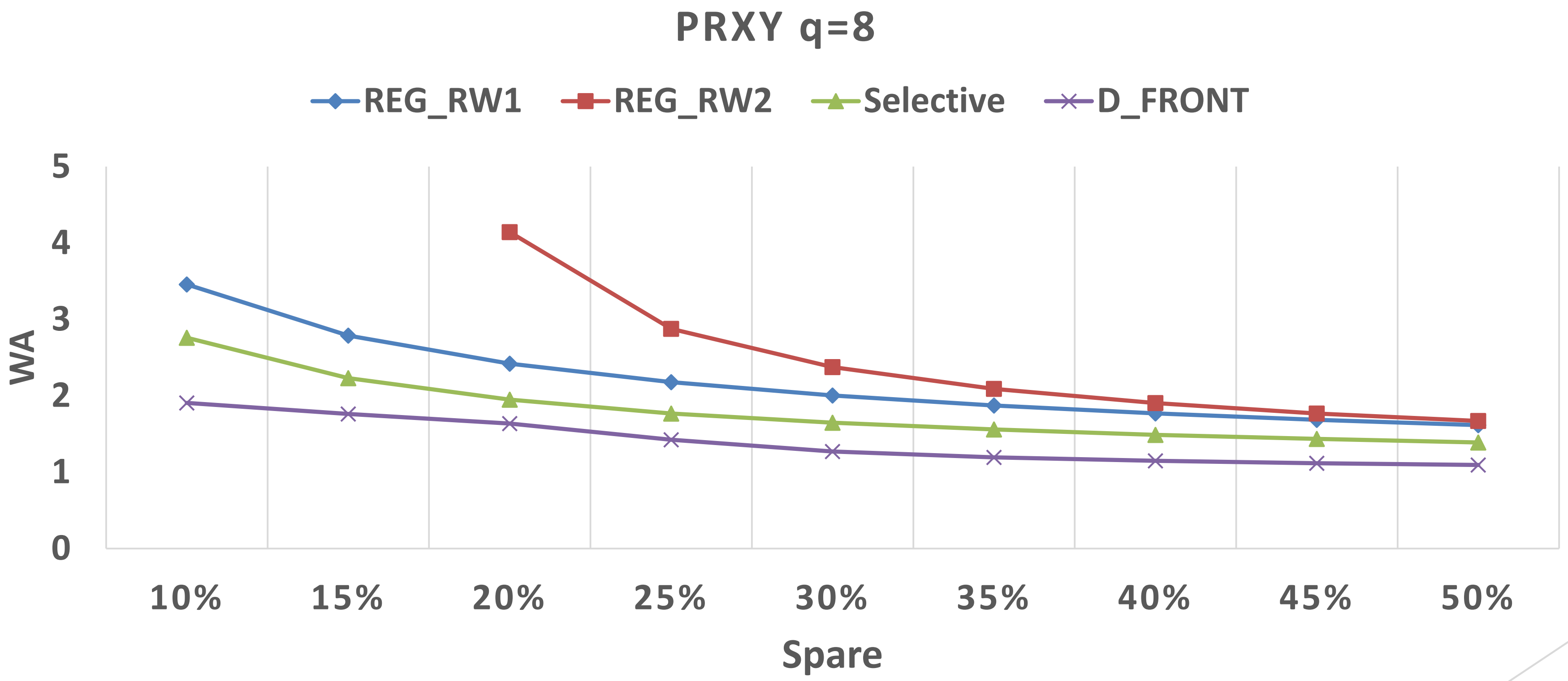


Evaluation

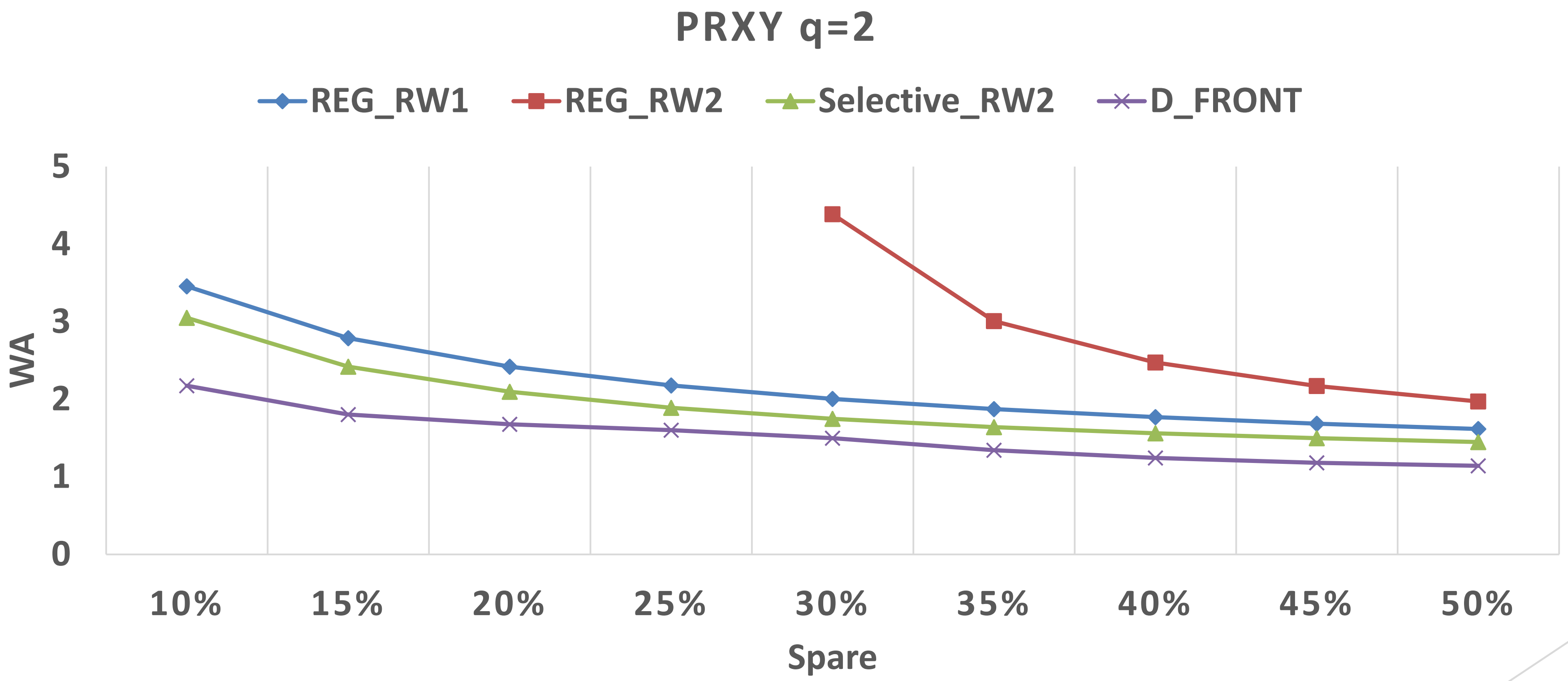


SNIA
IOTTA Repository
Advancing storage
REPOSITORY HOME
TRACES
FAQS
JOIN IOTTA
CONTRIBUTE
SNIA HOME
Tools
Traces
BLOCK I/O TRACES GO
NFS TRACES GO
SSSI WIOCP METRICS GO
SYSTEM CALL TRACES GO
HISTORICAL SECTION GO
TOOLS GO
I WANT TO
Choose One
Home » Traces » Block I/O Traces » MSR Cambridge Traces
MSR Cambridge Traces
The following traces are free to download under the terms of the SNIA Trace Data Files Download License. Please note that cookies must be enabled within your browser in order to download traces.
For questions about downloading using a shell script, see Using Shell Scripts, and for more information about downloading using a Windows batch script, see Using Batch Scripts.
View Additional Columns
Trace Name
Details
Actions
MSR Cambridge Traces 1
1-week block I/O traces of enterprise servers at Microsoft
Long Description
Download Sample
MSR Cambridge Traces 2
1-week block I/O traces of enterprise servers at Microsoft
Long Description
Download Readme

Trace Results - TLC

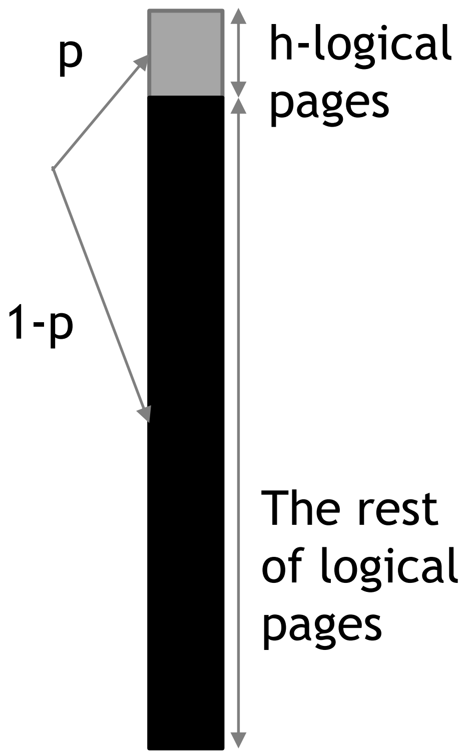


Trace Results - SLC

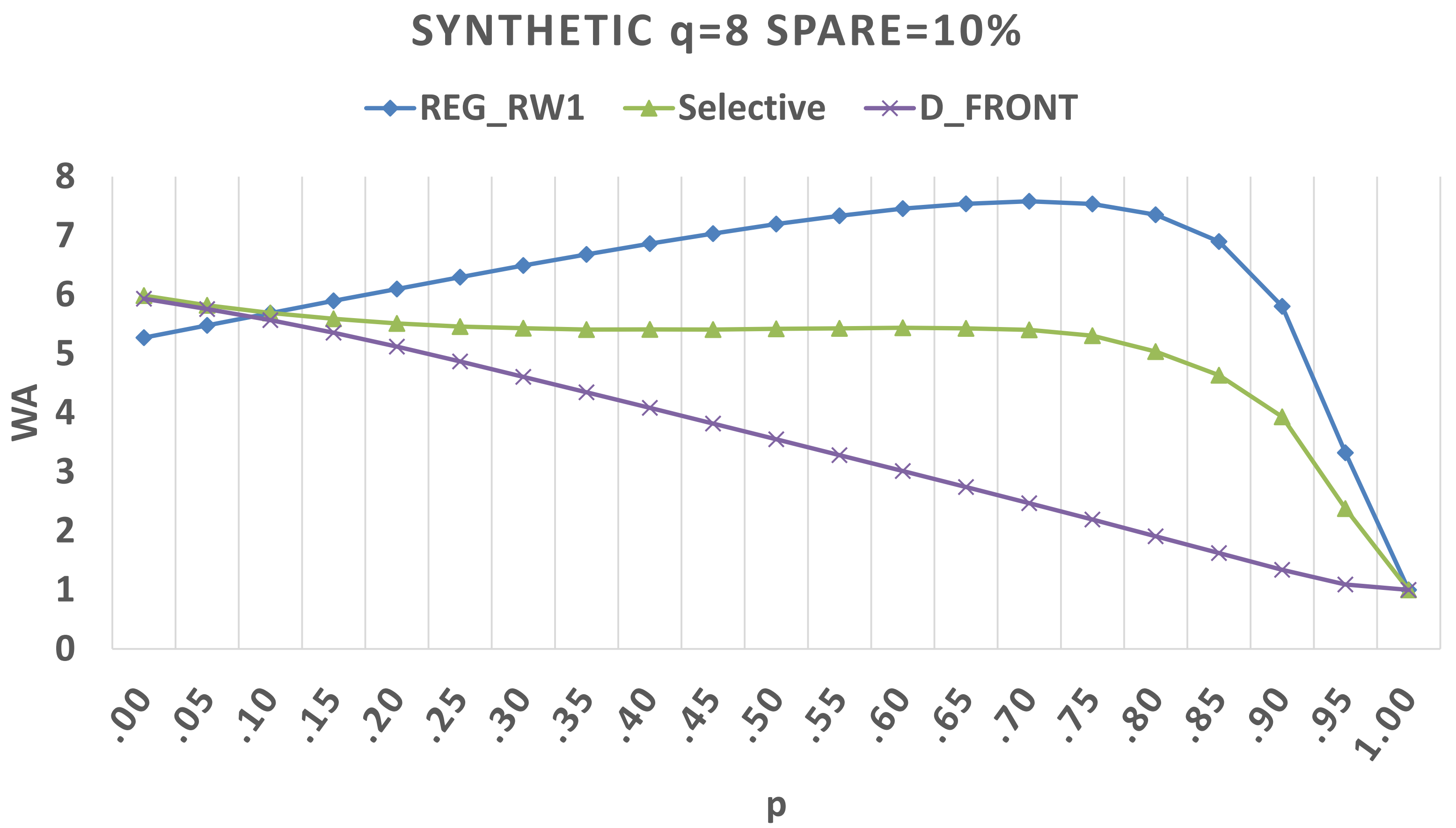


Synthetic Workloads: p-Locality

- p – the probability for a "hot" write
  - Write is chosen uniformly from a pool of hot pages
  - h – the size of the hot pool
- 1-p – the probability for a "cold" write
  - Write will be chosen from the rest of the pages
  - The "cold" page is turned to "hot"



p-Locality Results



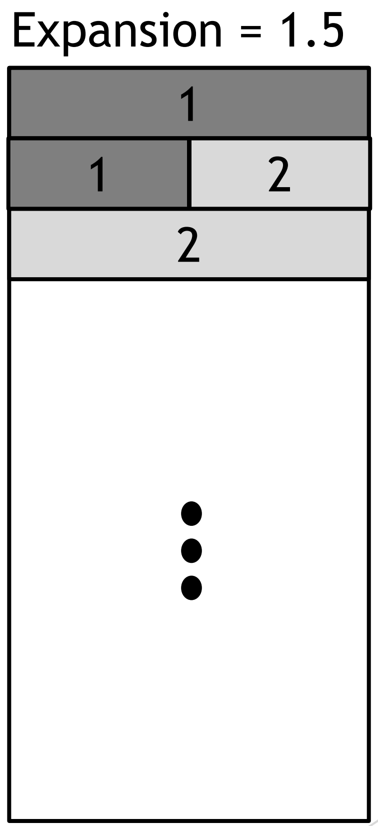
Real Implementation

- Problem: Variable Page Size
  - some pages use t=2 (larger).
  - some use t=1 (smaller).
- Solution:
  - w/o flash vendors support
  - w/   flash vendors support

w/o Flash Vendors Support 1

- Normal Page Allocation
  - Normal pages – no change
  - Expanded pages – grouped together

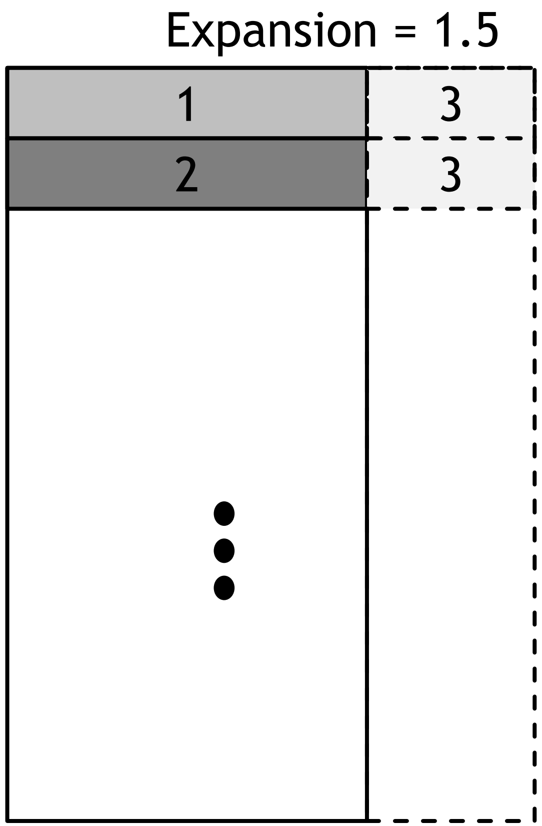
- Read penalty
- In-place write → RMW



w/o Flash Vendors Support 2

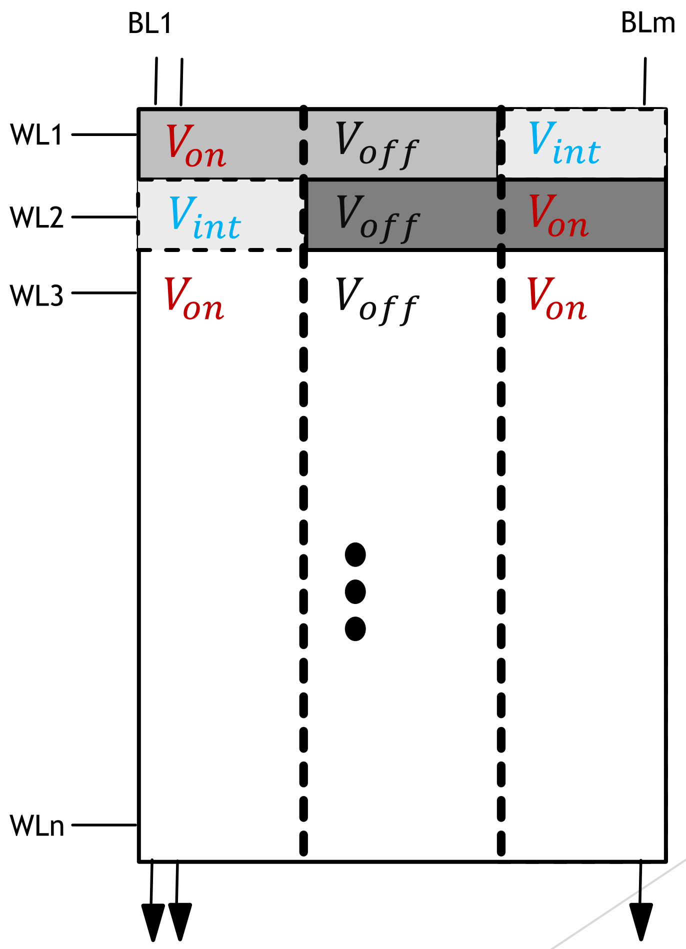
- Expanded Page Allocation
  - Expanded pages – no change
  - Normal pages – grouped together

- GC pages are grouped and written together
  - Write (Scatter) - no penalty
  - Read (Gather) – read penalty

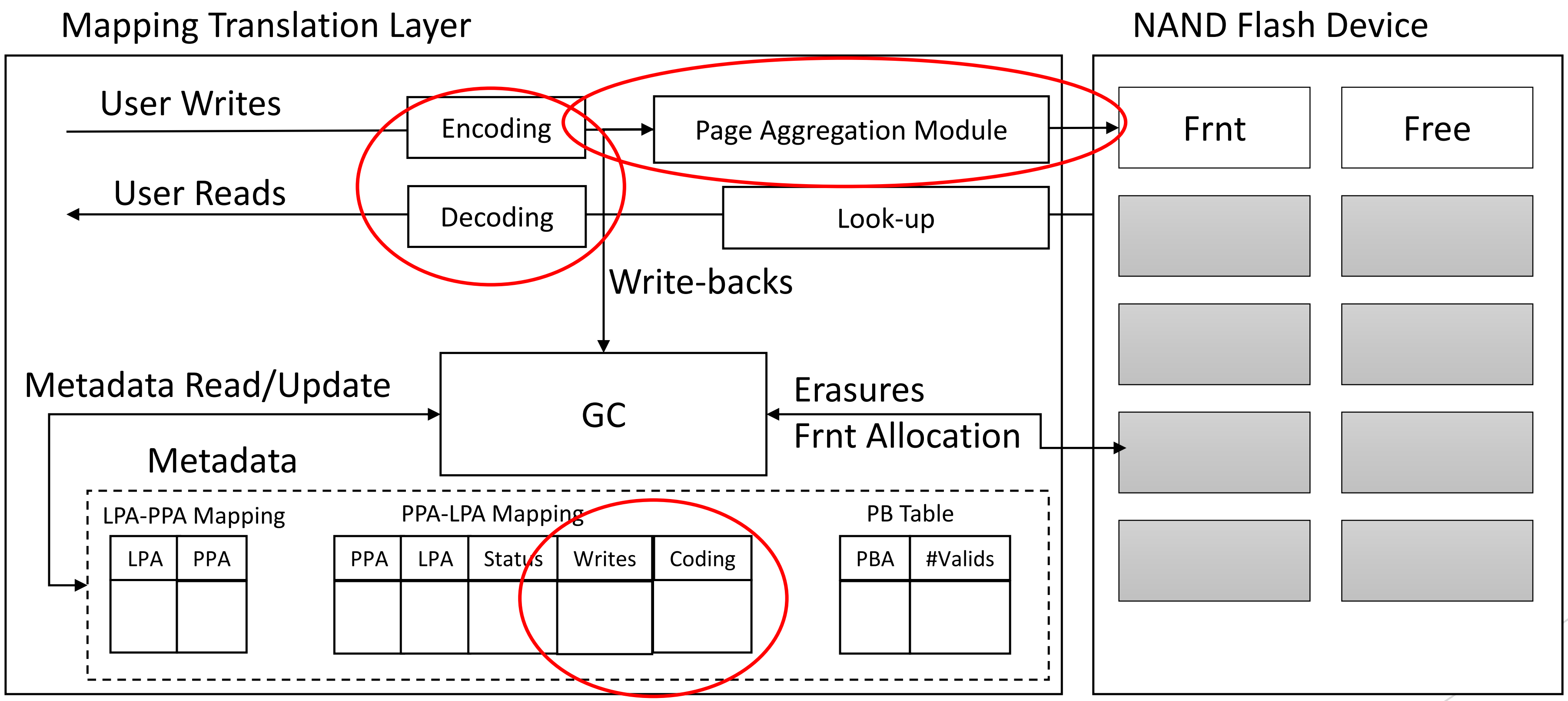


w/ Flash Vendors Support

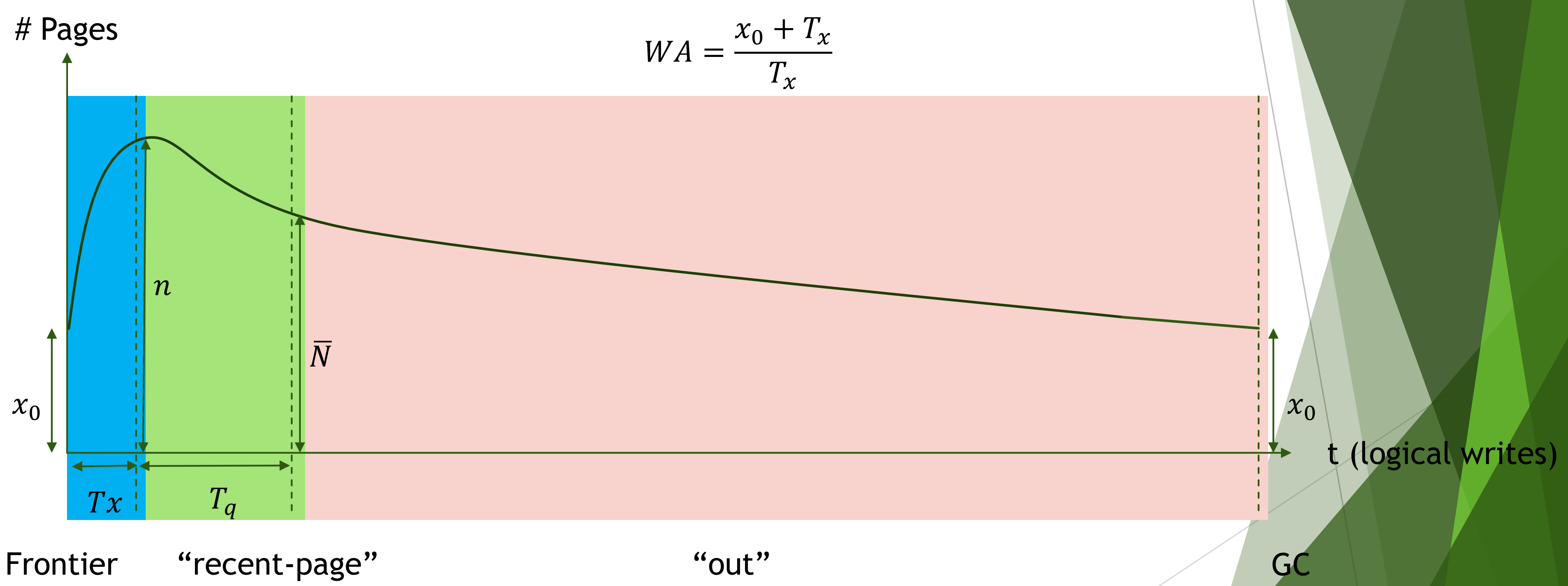
- Split pages are shifted
  - Parallel circuit read
- **No** penalty/ overhead
- Vendor support: Split word-line read



Device Diagram



The Block Life Cycle - Analysis



Conclusion

- New scheme to improve WA with selective MW-writes
- Insensitive to specific workload properties
- Good potential for SSD deployment