How to Enhance the Performance of SSDs by Coding Solutions?

Eitan Yaakobi

Technion
Israel Institute of Technology

Flash Memory Structure

- A group of cells constitute a page
- A group of pages constitute a block
  - In SLC flash, a typical block layout is as follows
  - Typical page size is 2-16KB

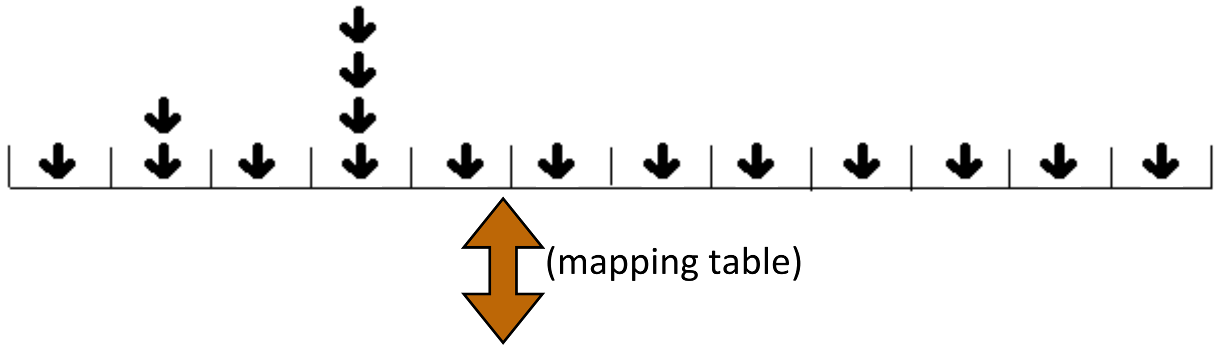
page 0	page 1
page 2	page 3
page 4	page 5
. . .	. . .
page 62	page 63

Flash Memory Structure

- Flash limitations:
  - Pages are written sequentially in the block
  - Erasures can be done only in the block level
- So how are pages written?
  - Sequentially one after the other into the physical blocks
  - Need a table to map between logical and physical addresses
  - Need to have Garbage Collection (GC) to support more writes

System Example of Writing

Logical Space:
(12 pages)



Physical Space:
(16 pages in 4 blocks)

Valid	Invalid	Valid	Valid
Invalid	Valid	Valid	Valid
Valid	Valid	Invalid	Valid
Valid	Invalid	Valid	Valid

Initial condition: Start with an empty memory

User writes uniformly and randomly distributed on user space

stationary condition: Logical memory is always full (worst case)

System

Garbage Collection

Logical Space:
(12 pages)

Physical Space:
(16 pages in 4 blocks)

Valid	Invalid	Valid	Valid
Invalid	Valid	Valid	Valid
Valid	Valid	Invalid	Valid
Valid	Invalid	Valid	Valid

Time to erase

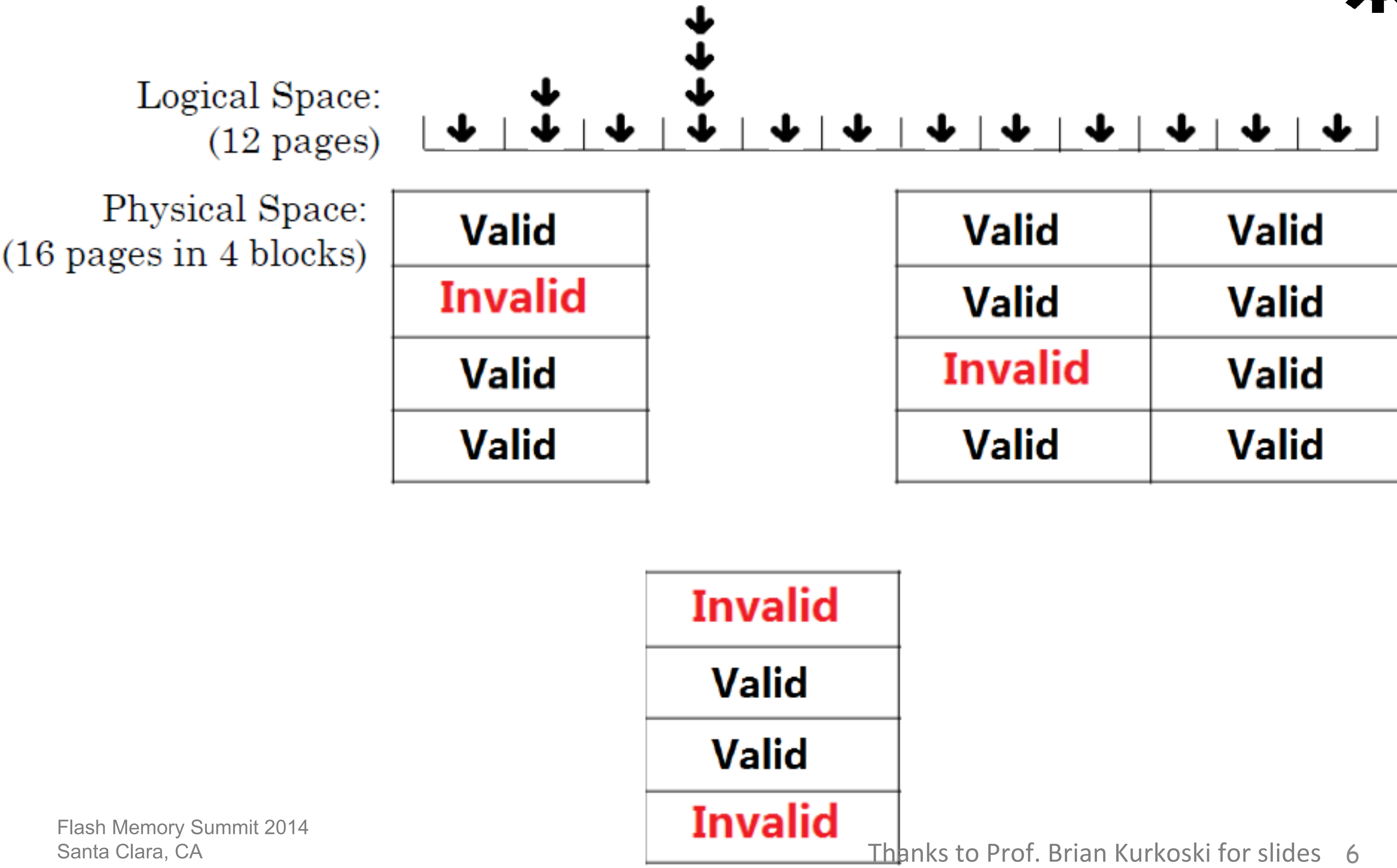
Greedy Garbage collection:

- Block with most invalid pages

Only two writes needed

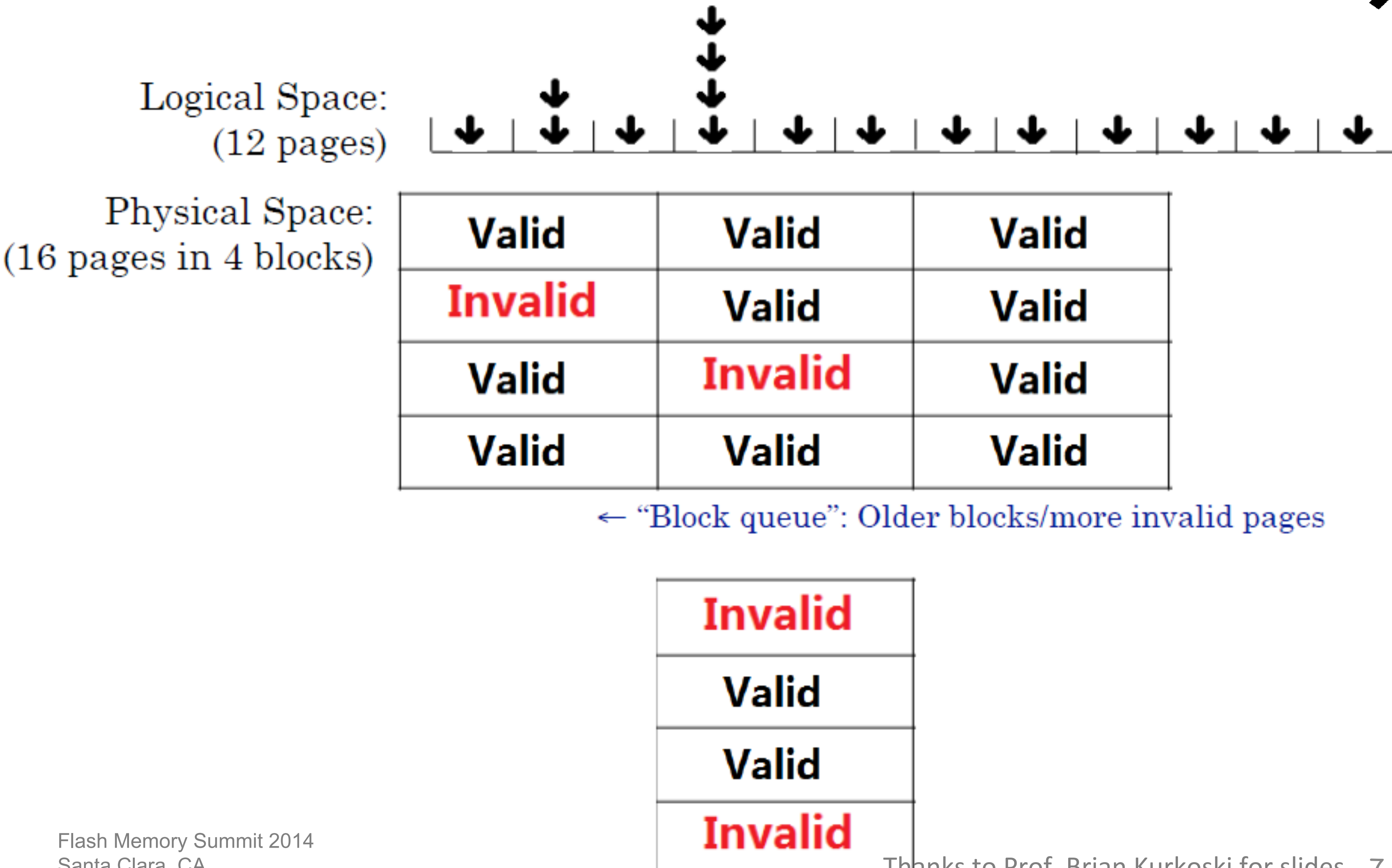
System

Garbage Collection



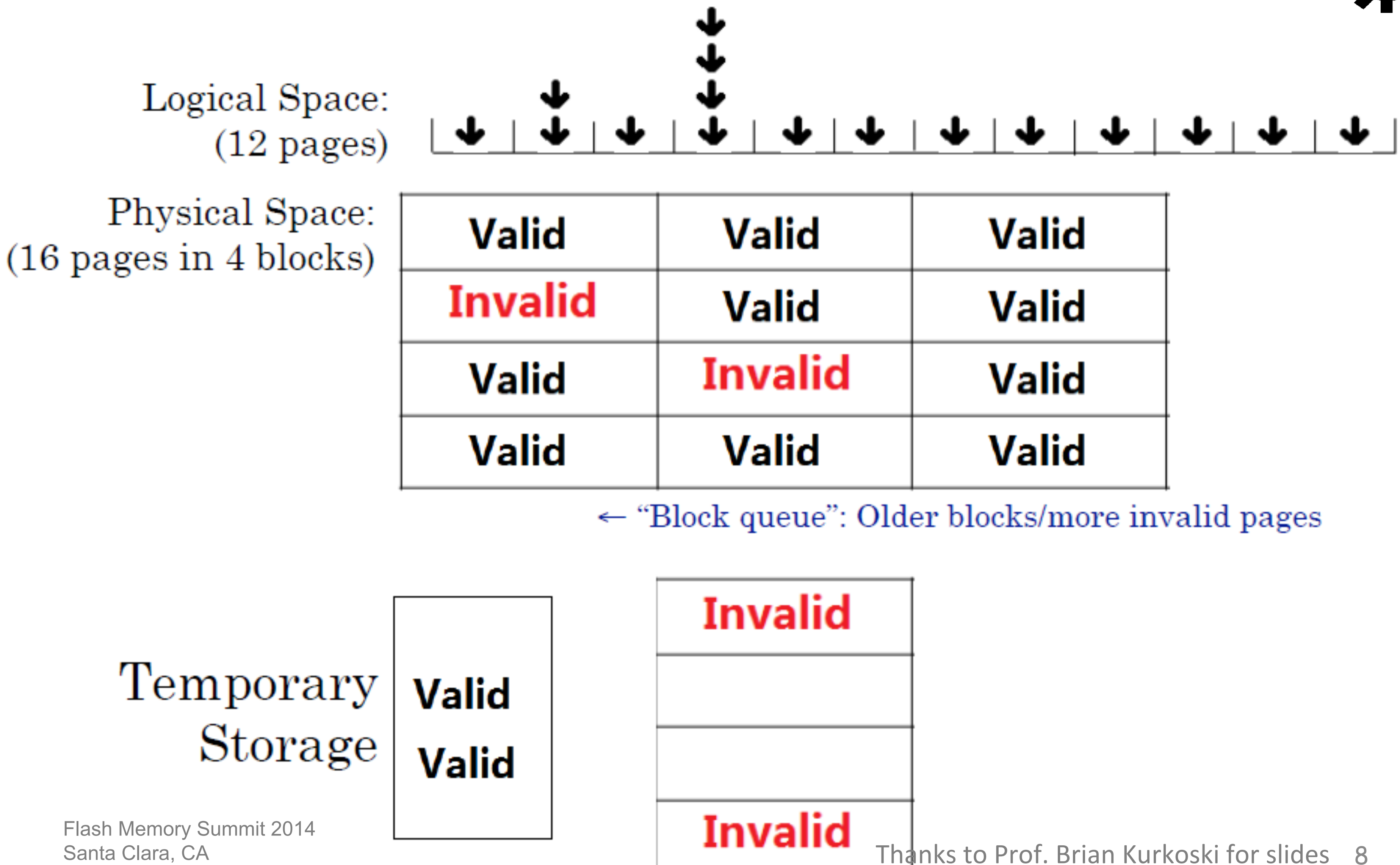
System

Garbage Collection

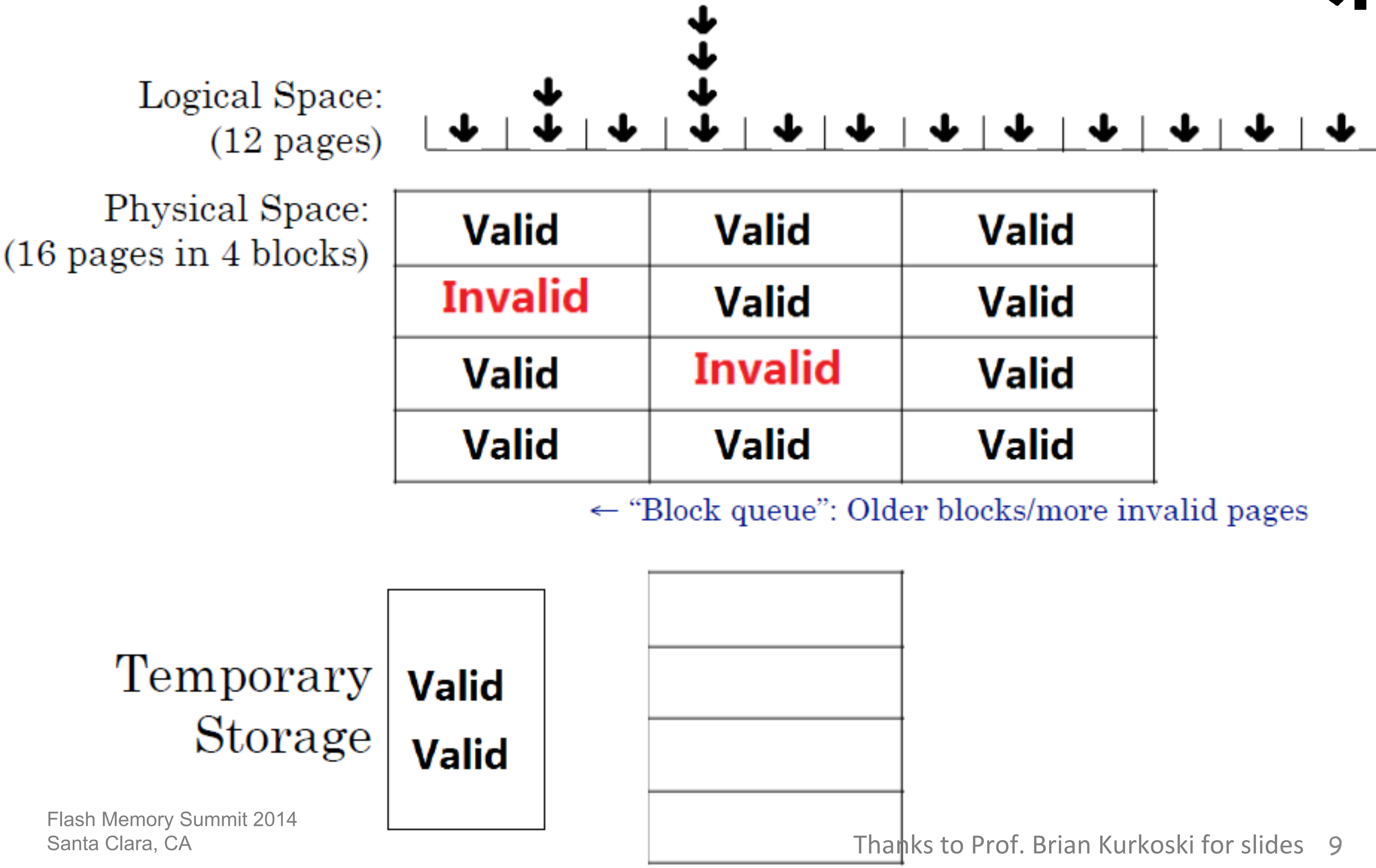


System

Garbage Collection

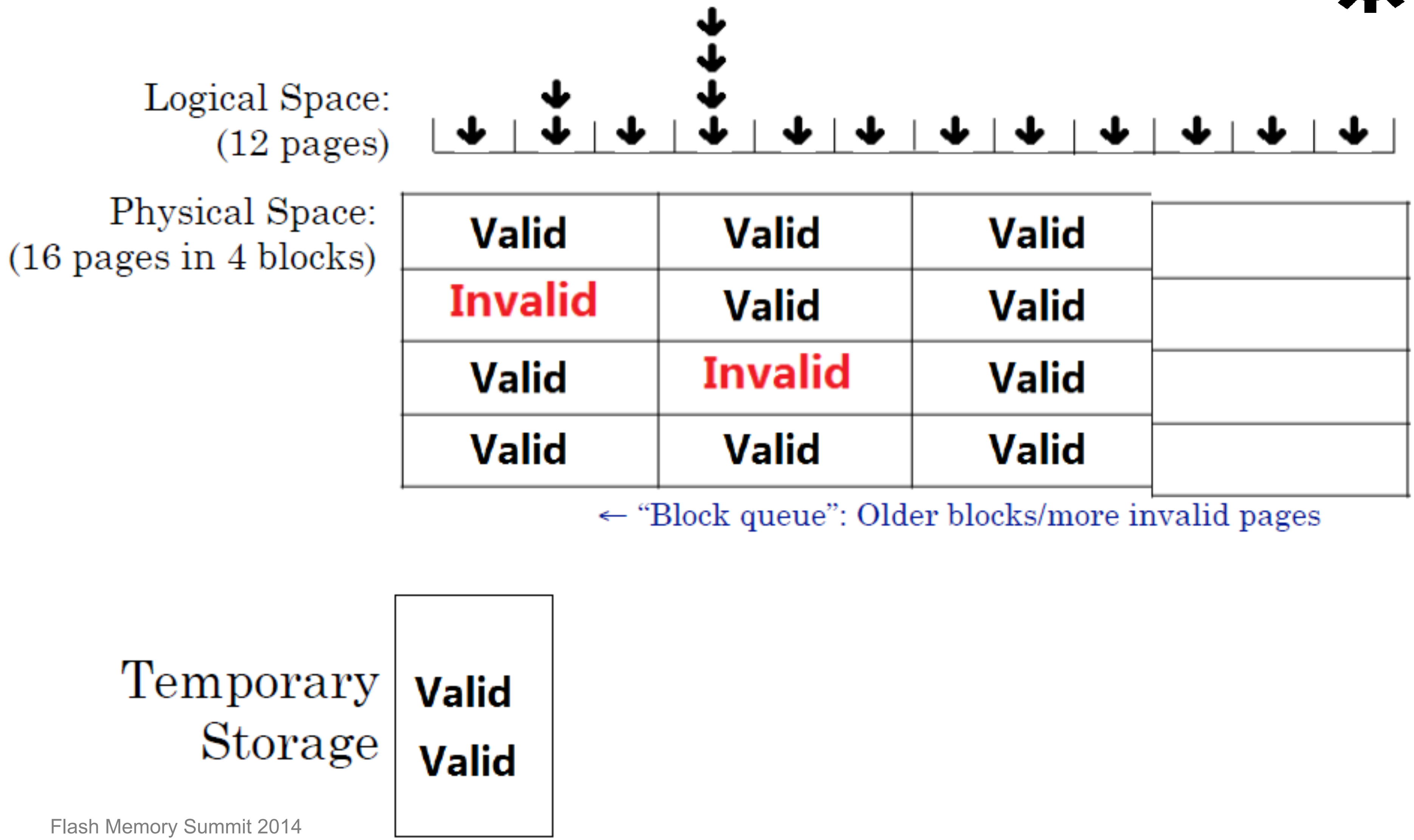


System
Garbage Collection



System

Garbage Collection



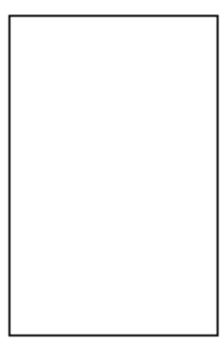
System
Garbage Collection

Logical Space:
(12 pages)

Physical Space:
(16 pages in 4 blocks)

Valid	Valid	Valid	
Invalid	Valid	Valid	Valid
Valid	Invalid	Valid	Valid
Valid	Valid	Valid	

Temporary Storage



Time to erase

Greedy Garbage collection:

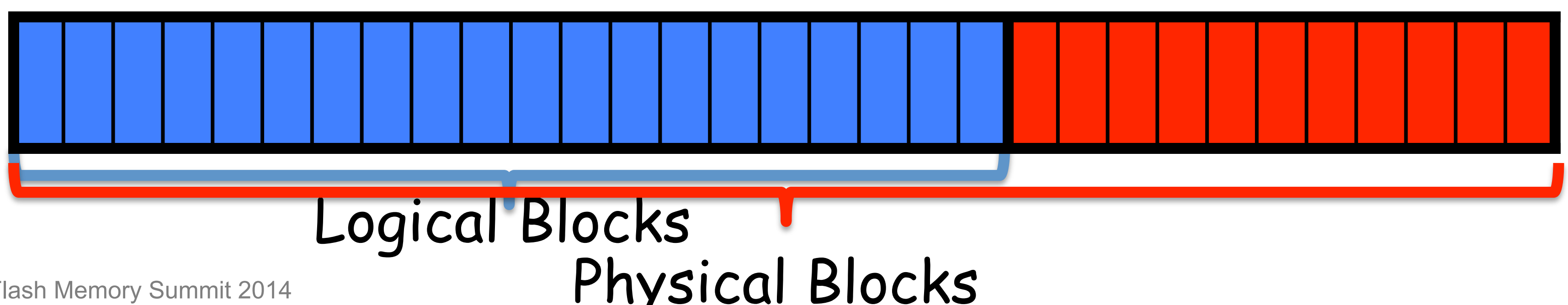
- Block with most invalid pages

Only two writes needed

Thanks to Prof. Brian Kurkoski for slides

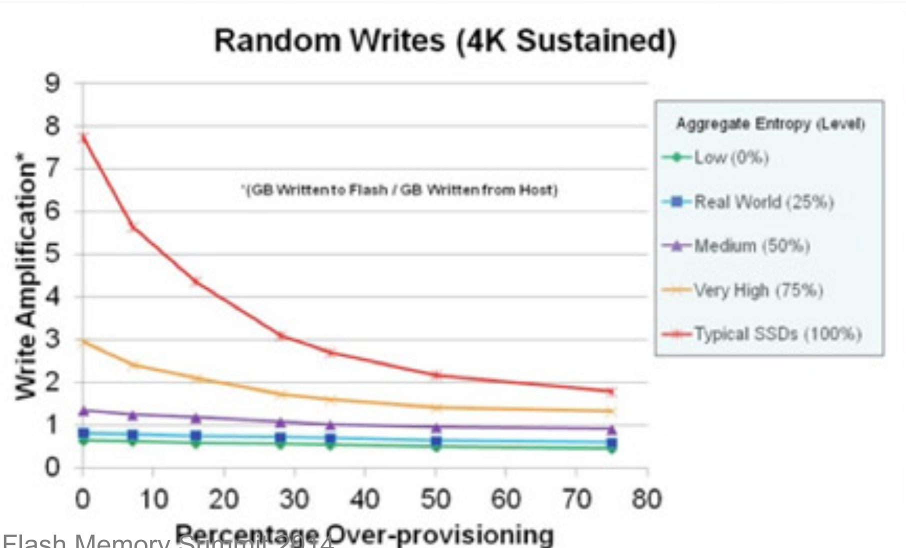
Greedy Garbage Collection

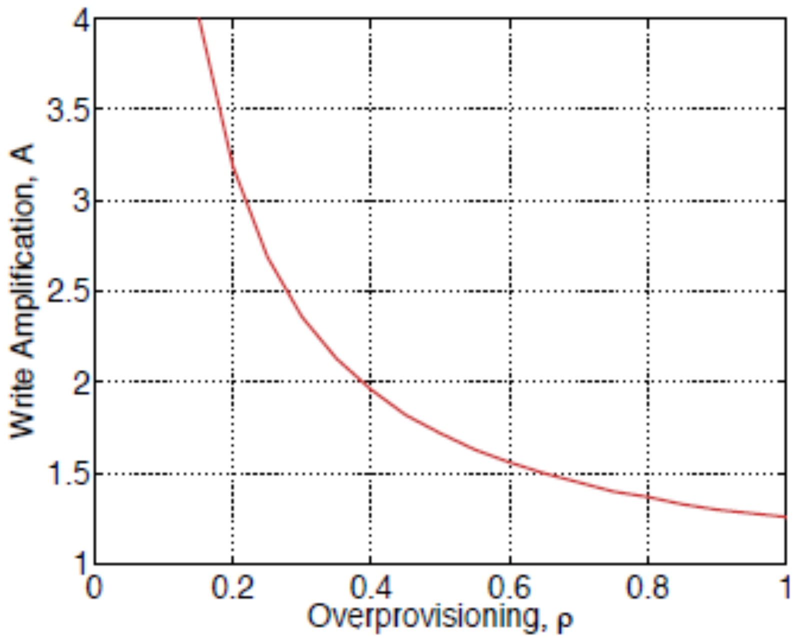
- **Write amplification** = $\frac{\#\ \text{Physical writes}}{\#\ \text{Logical writes}}$
- **Overprovisioning** = (T-U)/U ;
  T = #physical blocks, U = #logical blocks
  There is more physical than logical memory
- **Theorem**: Greedy garbage collection is optimal in order to reduce the write amplification (for uniform writing)



Analysis

- **Write amplification** = $\frac{\text{\# Physical writes}}{\text{\# Logical writes}}$
- **Overprovisioning** = (T-U)/U ;
  T = #physical blocks, U = #logical blocks
- **Question**: How are the overprovisioning factor and write amplification related?



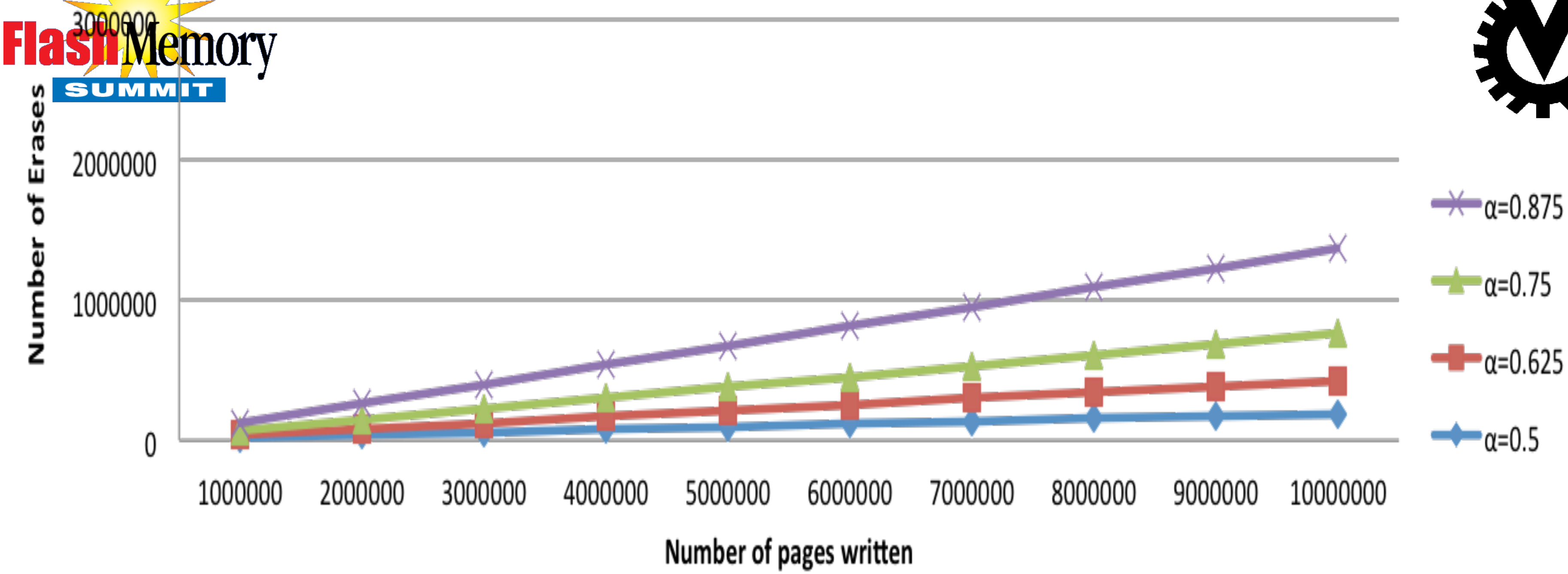


Analysis

- **Write amplification** = $\frac{\text{\# Physical writes}}{\text{\# Logical writes}}$
- **Overprovisioning** = (T-U)/U ;
  T = #physical blocks, U = #logical blocks
- **Question'**: How are the overprovisioning factor and write amplification related, ***under random uniform writing***?
  - **N** = #logical page writes ; **M** = # physical page writes
  - **E** = #block erasures = M/Z , **Z** = # pages in a block
- On average: **Y = α'Z** valid pages in an erased block
  - **M = N + EY**
  - E = M/Z = (N+EY)/Z; (Z-Y)E = N; E=N/(Z-Y);

$$E=N/Z(1-\alpha')$$

- **Question**: What's the connection b/w **α=U/T** and **α'=Z/Y**?
- **Answer**: $\alpha = (\alpha'-1)/\ln(\alpha')$ (Menon '95, Desnoyers '12)



- **N** = #logical page writes ; **M** = # physical page writes
- **E** = #block erasures = M/Z , **Z** = # pages in a block

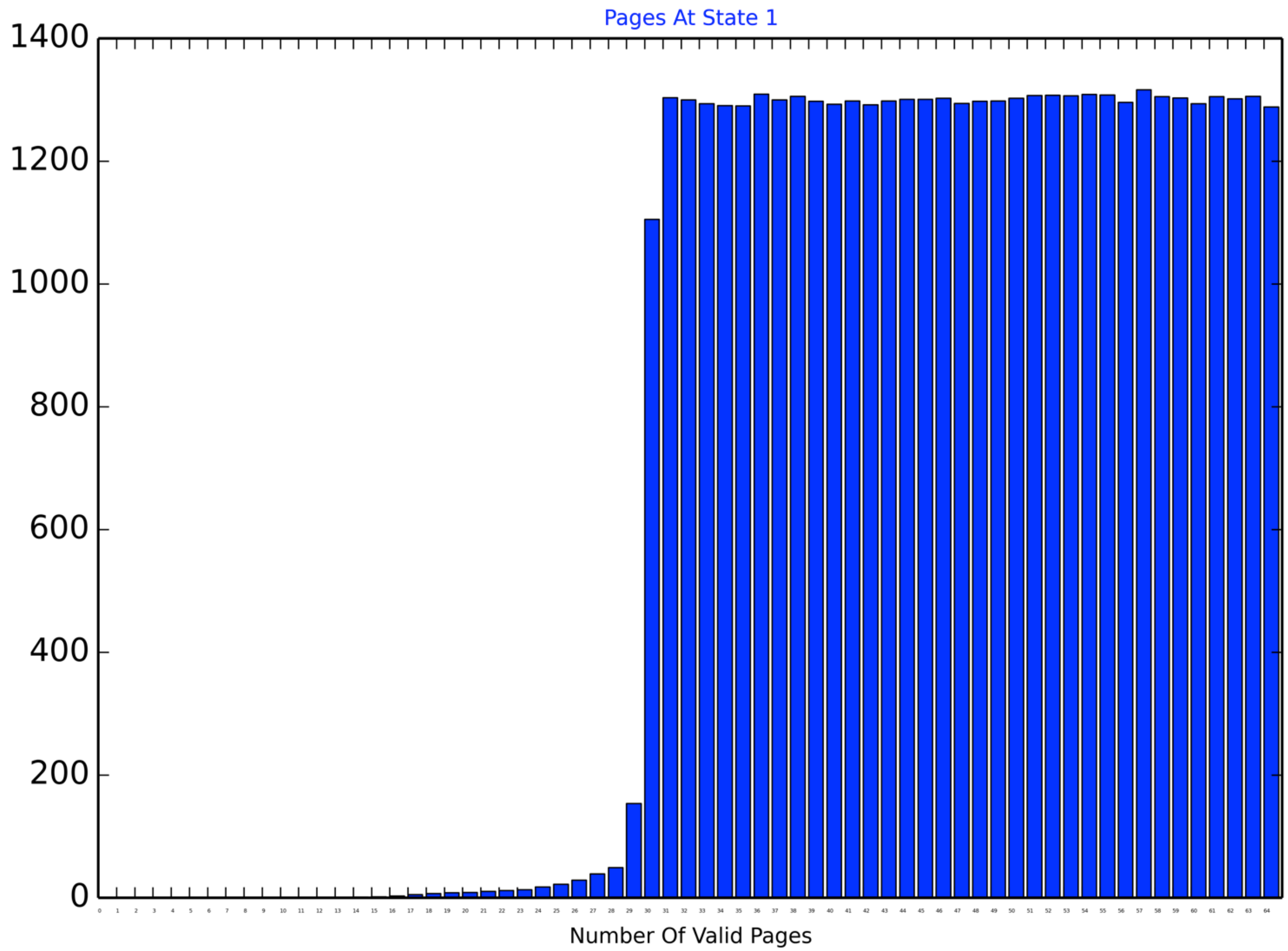
- On average: **Y = α'Z** valid pages in an erased block

$$E=N/Z(1-\alpha')$$

- **Question**: What's the connection b/w **α=U/T** and **α'=Z/Y**?
- **Answer**: $\alpha = (\alpha'-1)/\ln(\alpha')$ (Menon '95, Desnoyers '12)

Steady State Behavior

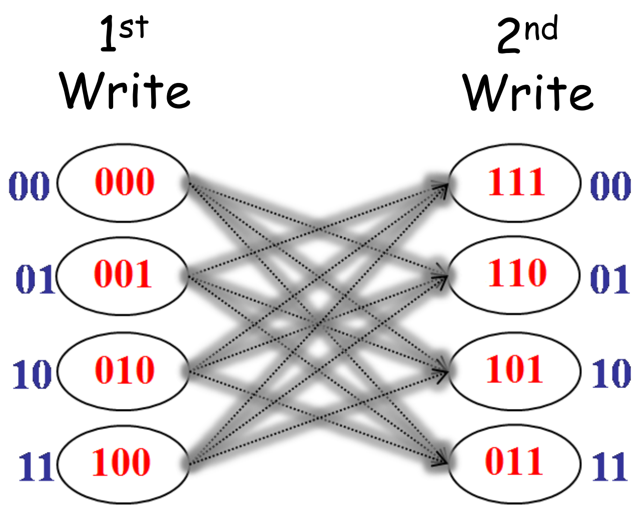
#blocks × #valid pages in block



Write-Once Memories (WOM)

- Introduced by **Rivest and Shamir**, "*How to reuse a write-once memory*", 1982
- The memory elements represent bits (2 levels) and are irreversibly programmed from **'0'** to **'1'**

Bits Value	1st Write	2nd Write
00	000	111
01	001	110
10	010	101
11	100	011



Write-Once Memories (WOM)

- Examples:

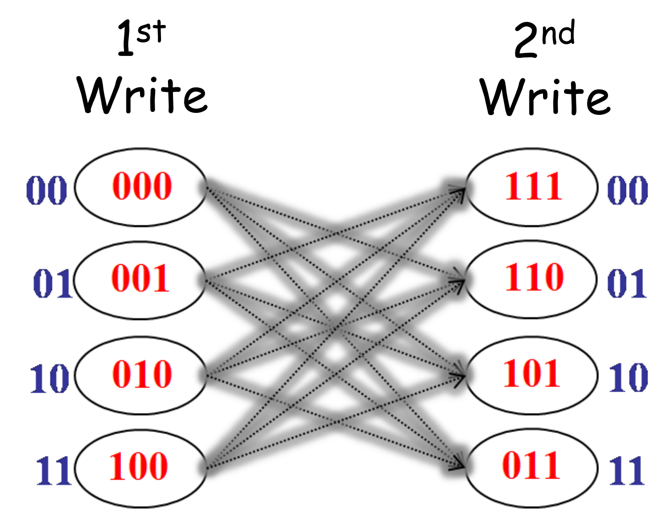
data	Memory State

data	Memory State

data	Memory State

data	Memory State

Bits Value	1st Write	2nd Write
00	000	111
01	001	110
10	010	101
11	100	011



Write-Once Memories (WOM)

- Introduced by **Rivest and Shamir**, "*How to reuse a write-once memory*", 1982
- The memory elements represent bits (2 levels) and are irreversibly programmed from **'0'** to **'1'**

Bits Value	1st Write	2nd Write
00	000	111
01	001	110
10	010	101
11	100	011

Q: How many cells to write **100** bits **twice**?

P1: Is it possible to do **better**…?

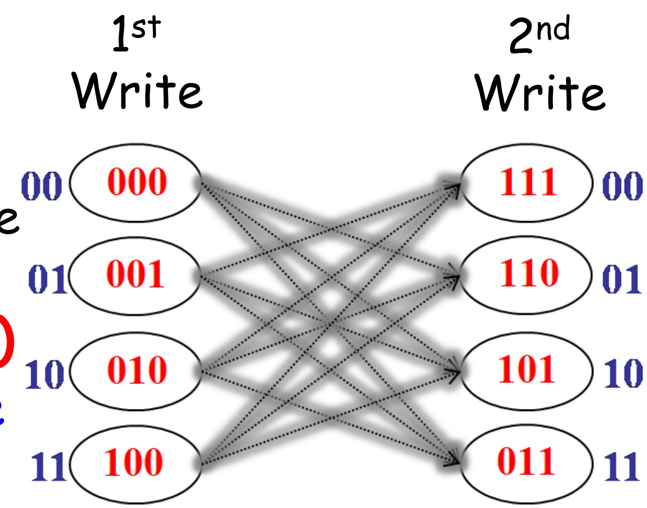
P2: How many cells to write **k** bits **twice**?

P3: How many cells to write **k** bits **t** times?

P3': What is the total number of bits that is possible to write in **n** cells in **t** writes?

The max ***sum-rate*** of a t-write code is **log(t+1)**

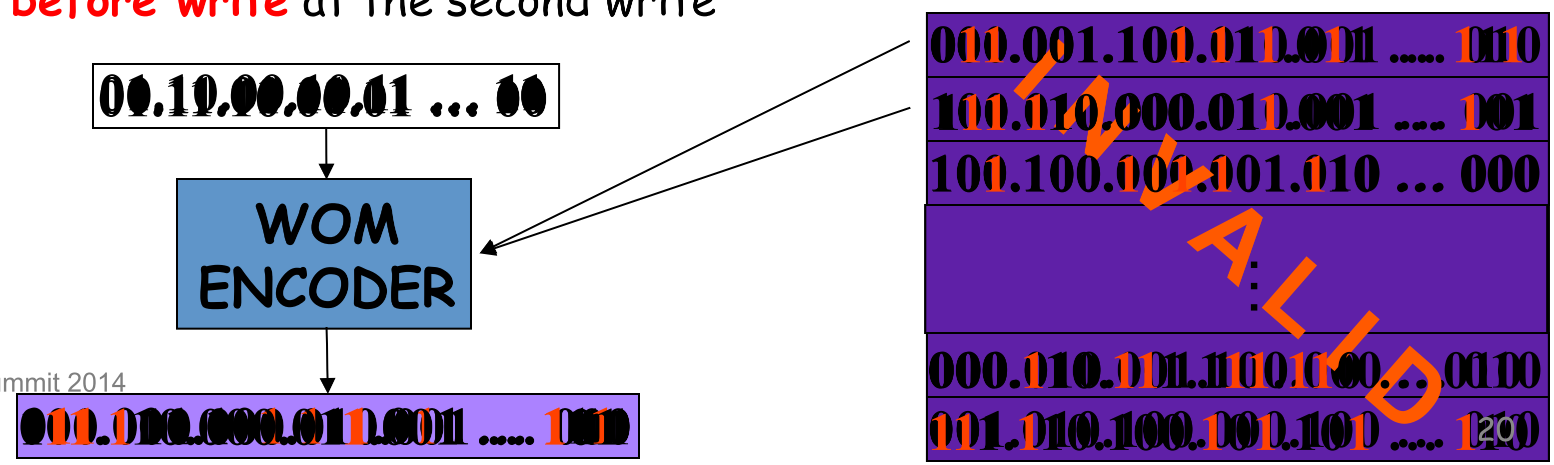
Several constructions aiming for high sum-rate



WOM Implementation in SLC

- A scheme for storing two bits twice using only three cells **before erasing the cells**
- The cells only **increase** their level
- How to implement? (in **SLC** block)
  - Each page stores **2KB/1.5 = 4/3KB** per write
  - A page can be written twice before erasing
  - Pages are **encoded** using the **WOM code**
  - When the block has to be rewritten, mark its pages as **invalid**
  - Again write pages using the WOM code **without erasing**
  - **Read before write** at the second write

data	1st write	2nd write
00	000	111
01	100	011
10	010	101
11	001	110



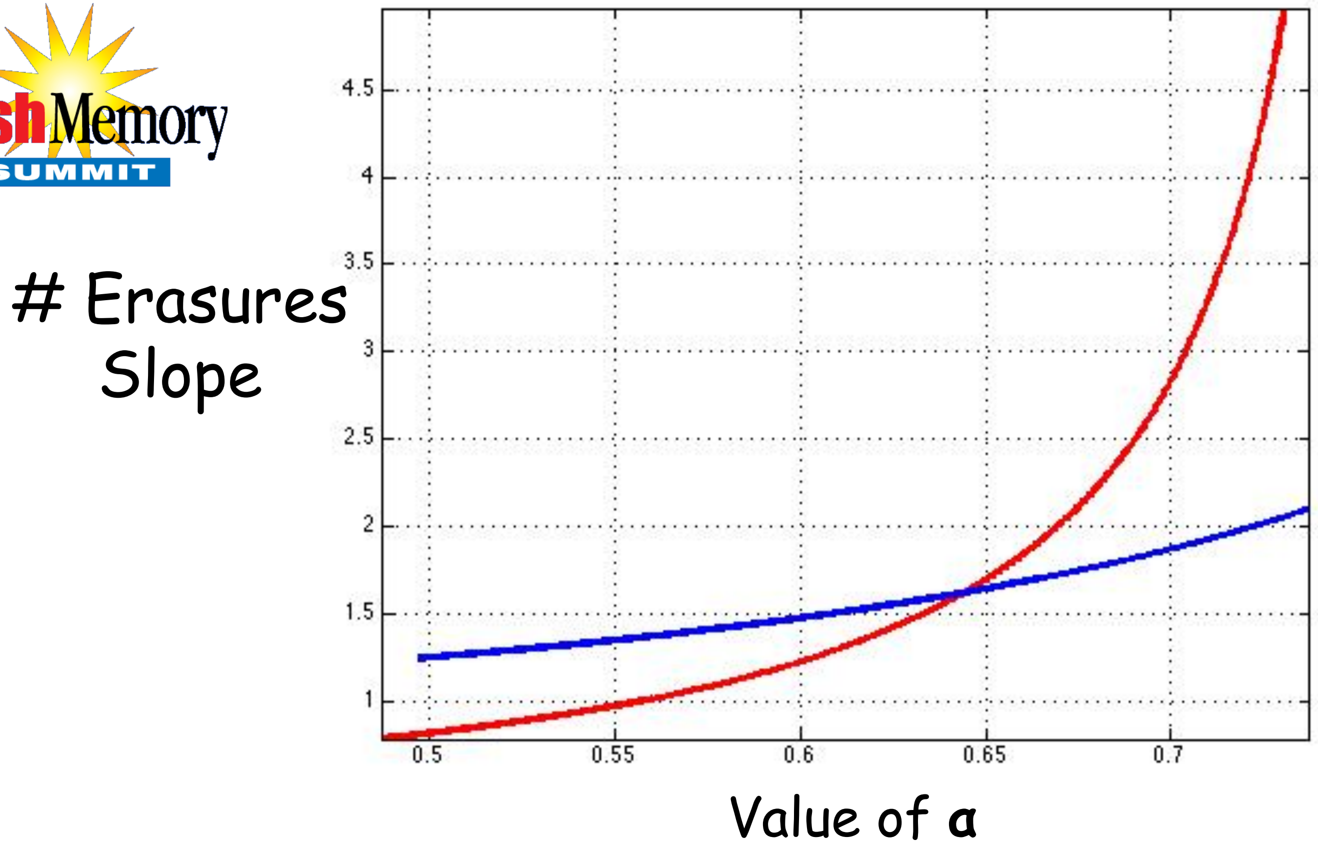
Why/When to Use WOM Codes?

- **Disadvantage**: sacrifice a large amount of the capacity
  - **Ex**: Two write WOM codes
    - The best sum-rate is **log3≈1.58**
    - Can write (at most) only **0.79n** bits so there is a lost of (at least) **21%** of the capacity
- **Advantage**: Can increase the lifetime of the memory and reduce the write amplification

Analysis with WOM

- What is # of block erasures when using a WOM code?
  - Assume one uses a 2-write WOM code with rate $R = 0.77$
  - Repeat the same algorithm of Greedy GC
  - $\beta = U'/T'$ =ratio b/w logical and physical blocks
  - On average, $Y_2=\beta' Z$ valid pages in an "erased" block
  - $\beta = (\beta'-1)/\ln(\beta')$
  - $E_1 = M/Z = (N+E_1 Y)/Z$; $E_1=N/Z(1-\alpha')$
  - $E_2= M/2Z=(N+E_2 Y_2)/2Z$; $E_2=N/2Z(1-\beta')$
  - **$\alpha = R\beta$**: to have the same amount of logical and physical memory
  - It is better to use 2-write WOM iff
    $N/Z(1-\alpha') = E_1 > E_2 = N/2Z(1-\beta')$ iff
    $\beta' < (1+\alpha')/2$ iff $\alpha' < 0.385$ iff **$\alpha < 0.6443$**

    Over-provisioning ratio = 0.55

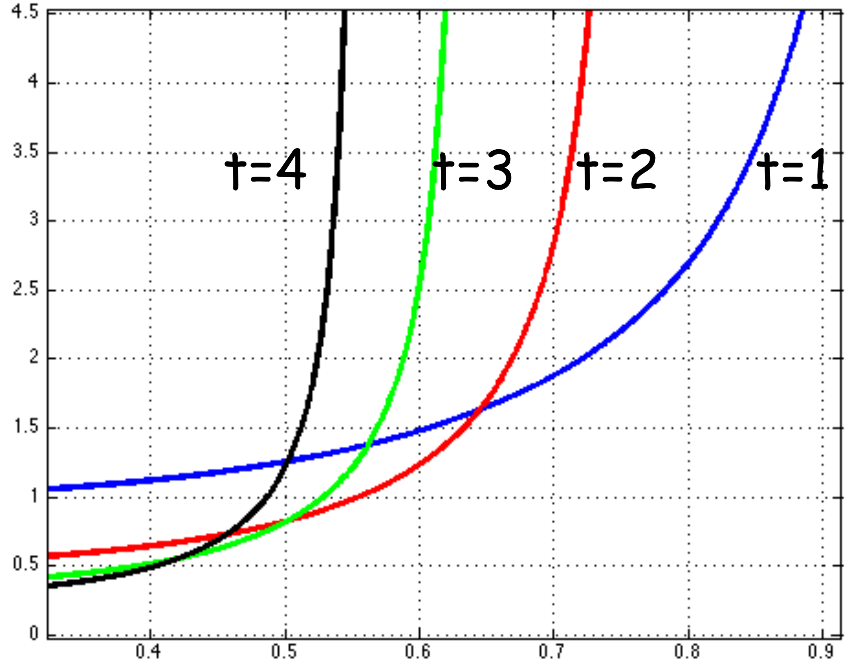


- On average, $Y_2=\beta' Z$ valid pages in an "erased" block
- $E_1 = M/Z = (N+E_1Y)/Z$; $E_1=N/Z(1-\alpha')$
- $E_2= M/2Z=(N+E_2Y_2)/2Z$; $E_2=N/2Z(1-\beta')$
- It is better to use 2-write WOM iff
  $\beta' < (1+\alpha')/2$ iff $\alpha' < 0.385$ iff **$\alpha < 0.6443$**

  Over-provisioning ratio = 0.55

Multi-write WOM Codes

- Similar Analysis for more than two writes
  - t=3: **α < 0.562** ➔ Over-provisioning ratio = 0.77
  - t=4: **α < 0.502** ➔ Over-provisioning ratio = 0.99

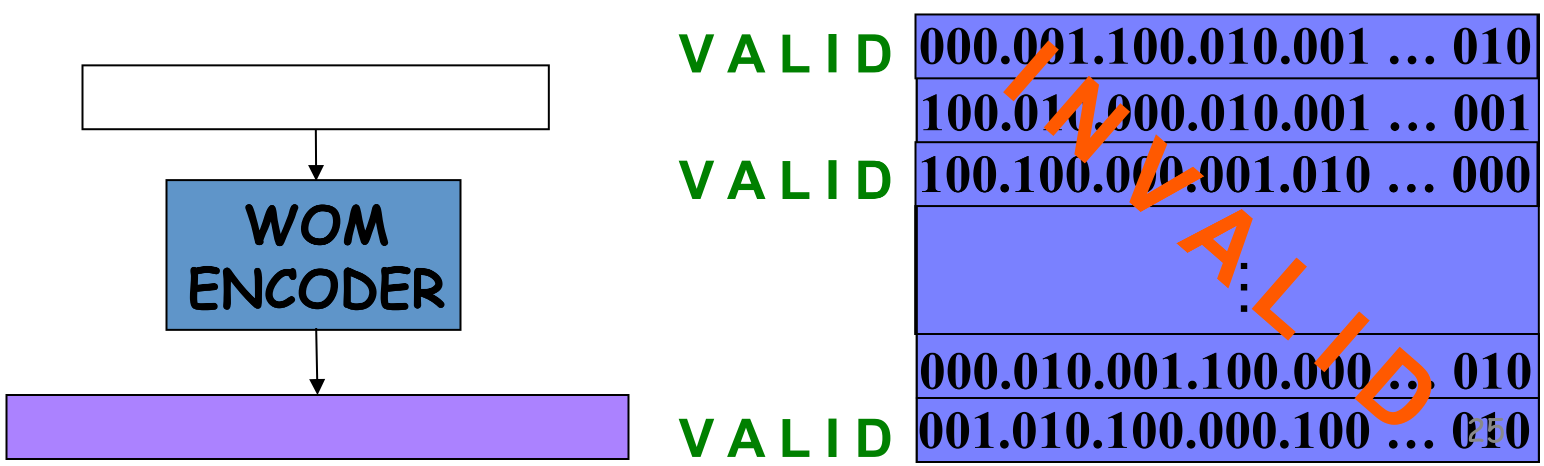


Value of α

WOM Implementation in SLC

- How to implement? (in **SLC** block)
  - Each page stores **2KB/1.5 = 4/3KB** per write
  - A page can be written twice before erasing
  - Pages are **encoded** using the **WOM code**
  - When the block has to be rewritten, mark its pages as **invalid**
  - **Improvement**:
    - Valid pages **are not** marked as invalid
    - If they are later updated, then this update is done in place (for "free")

data	1^{st} write	2^{nd} write
00	000	111
01	100	011
10	010	101
11	001	110

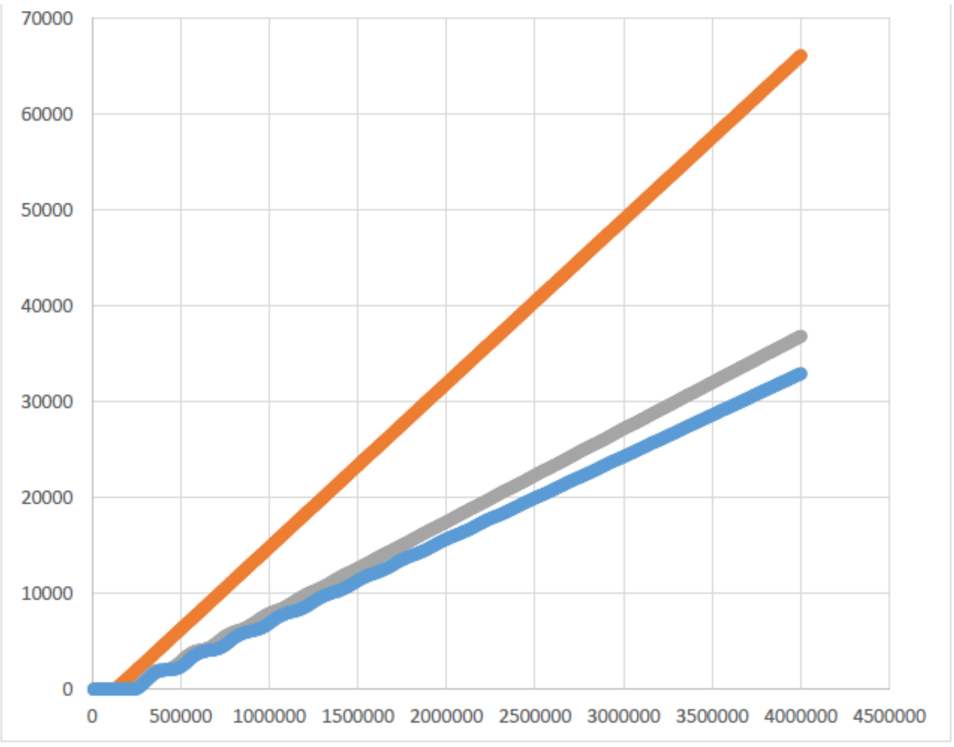


Example for α=0.5

Is it possible to do better?

When a block is moved from first to second state, the valid pages **are not** marked as invalid and if they are later on updated, then this update is done in place

Q1: How to change the GC algorithm?
Q2: What is the new threshold for α to get an improvement?



Summary

- Write Amplification analysis with WOM codes
  - WOM codes can help, but need to be careful when…
- **Ongoing work**
  - Improve the write amplification with minimum capacity penalty
  - Performance analysis: read and write speed
- **More areas of interest**
  - Channel model, modulation codes, Data compression, Wear leveling, 3D flash
  - PCM, Memristors, STT-MRAM, CB-RAM
  - Enterprise and RAID solutions for SSDs
- For more discussion – come to **BOOTH #803**!!!

Thank You!