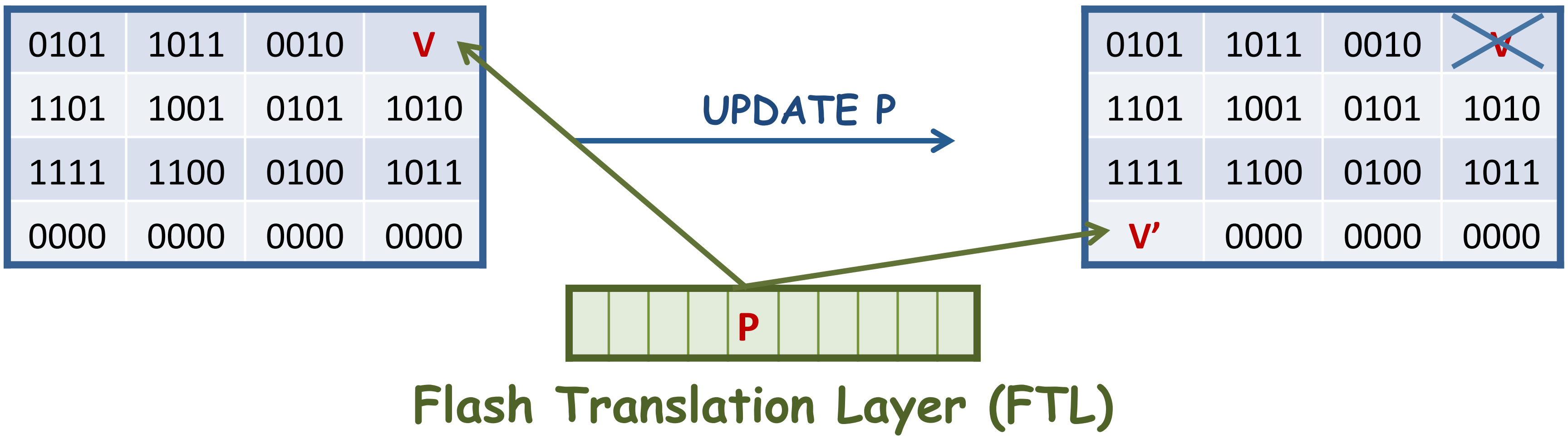
Write Once, Get 50% Free: Saving SSD Erase Costs Using WOM Codes

Eitan Yaakobi

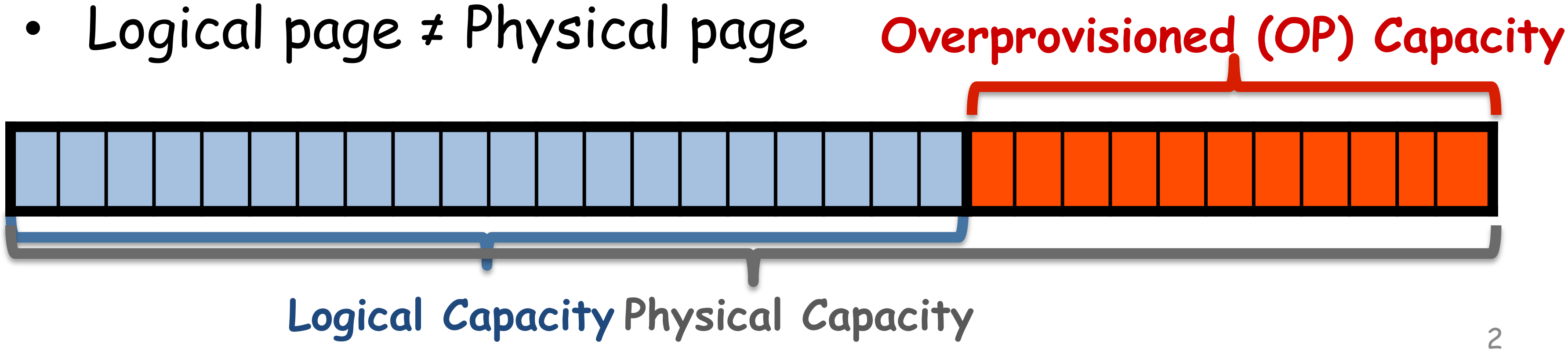
Joint work with:

Gala Yadgar, Alexander Yucovich, Gal Maor, and Assaf Schuster

Flash in a Nutshell



- **Out of place writes** replace updates
- Logical page ≠ Physical page



Flash in a Nutshell

~~0101~~	A	~~0010~~	B
~~1101~~	~~1001~~	~~0101~~	~~1010~~
~~1111~~	~~1100~~	C	~~1011~~
~~0010~~	~~0011~~	~~0100~~	D

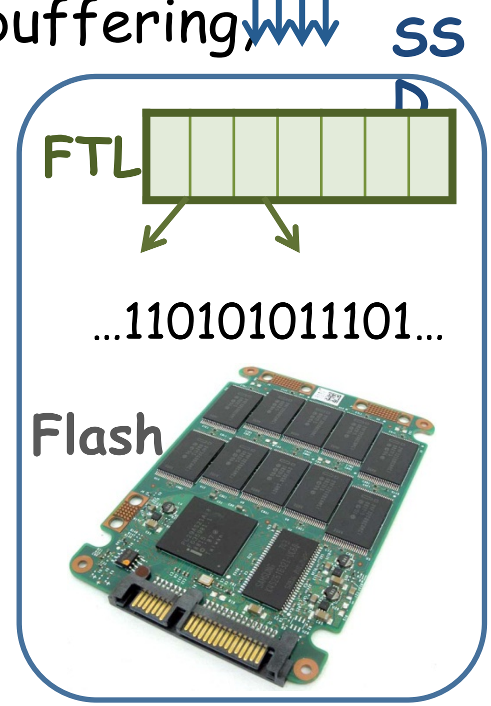
Garbage Collection →

A	B	C	D
0000	0000	0000	0000
0000	0000	0000	0000
0000	0000	0000	0000

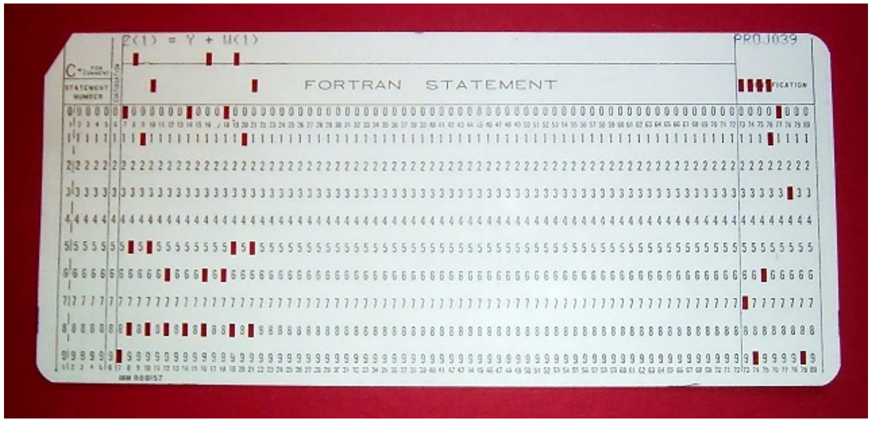
- **Garbage Collection (GC)** generates extra writes
- Write Amplification (WA) =
  (user writes + GC writes)/user writes
- Larger OP → Lower WA → Less erasures

How to extend SSD Lifetime?

- User level
  - Caching, admission control, specialized FS/DB…
- FTL
  - Wear leveling, throttling, partitioning, buffering, deduplication…
- Flash
  - Write voltage/speed…
- Code
  - Error correction
  - **WOM (write-once memory) codes overwrite without erasure**



Write-Once Memory Codes



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

(Rivest and Shamir, 1982)

- WOM Code: write ***n* bits** of information on ***m* cells**, *n>m*
- Example: **write *11*** and then **write *01***
  - Normally:

1	1	0	1

  - WOM code:

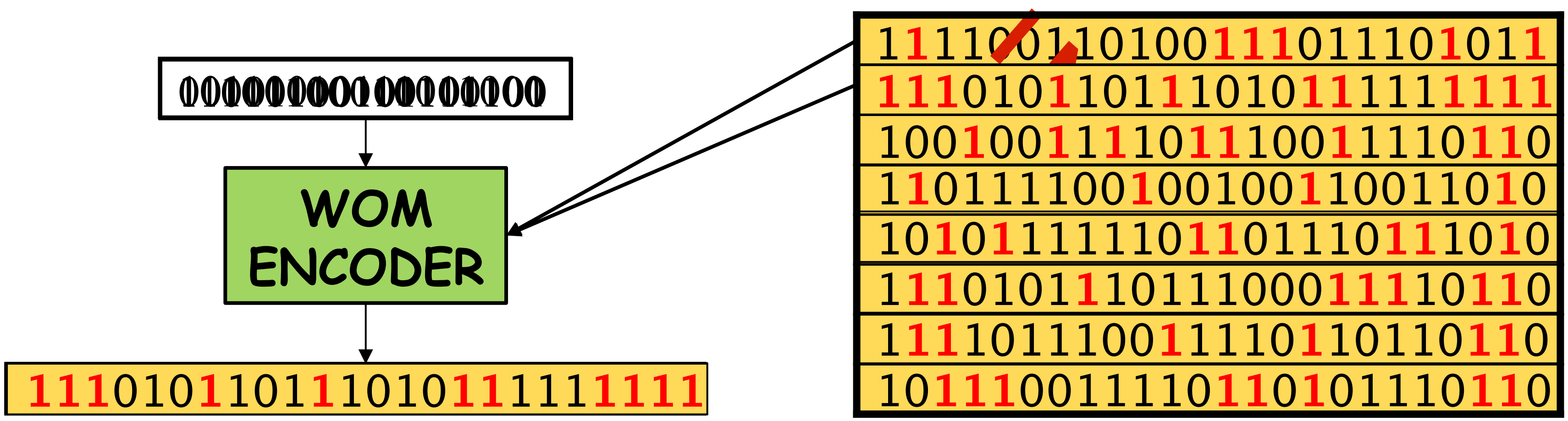
1	0	1

Typical Use of WOM Codes

- User writes **logical** data pages
- Page size **increases** with encoding
- Invalid pages are 'reused' **without erasing**
- **Read** before the second write

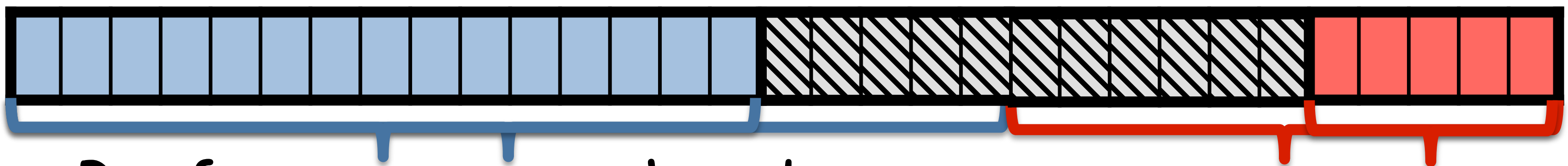
[Grupp et. al., MICRO '09][Jagmohan et.al., MSST '10]
[Loujie et. al., GLOBECOM '12]
[Jacobvitz et.al., HPCA '13][Odeh and Cassuto, MSST '14]

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



Drawbacks of Typical Use

- Capacity overhead:
  - 29%-50% **additional storage** is needed for WOM coding

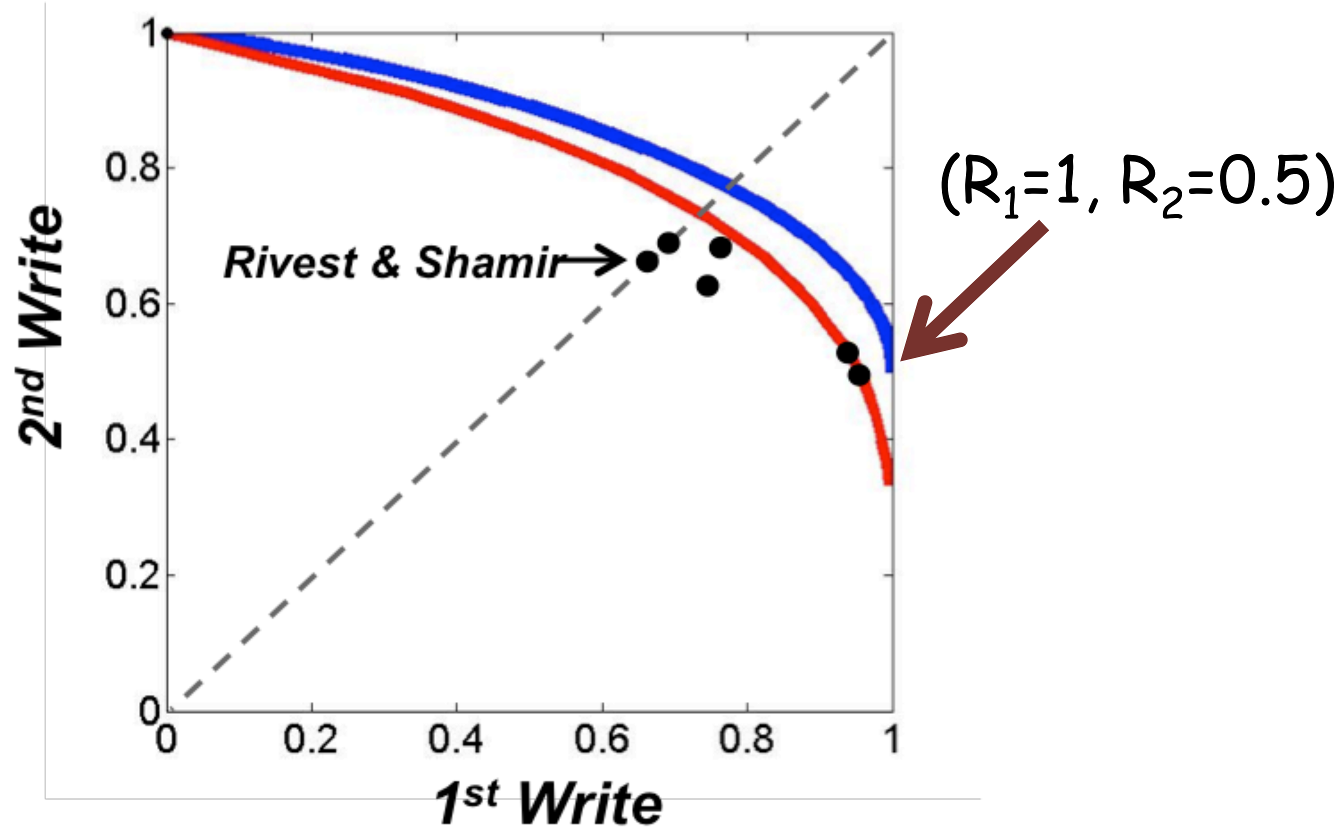


Overprovisioning

- Performance overheads:
  - I/O operations access 29%-50% **more bits**
  - A **read precedes** every second write
- Compatibility:
  - **Requires modification** in physical page size
  - Or access 2 physical pages

Our Approach

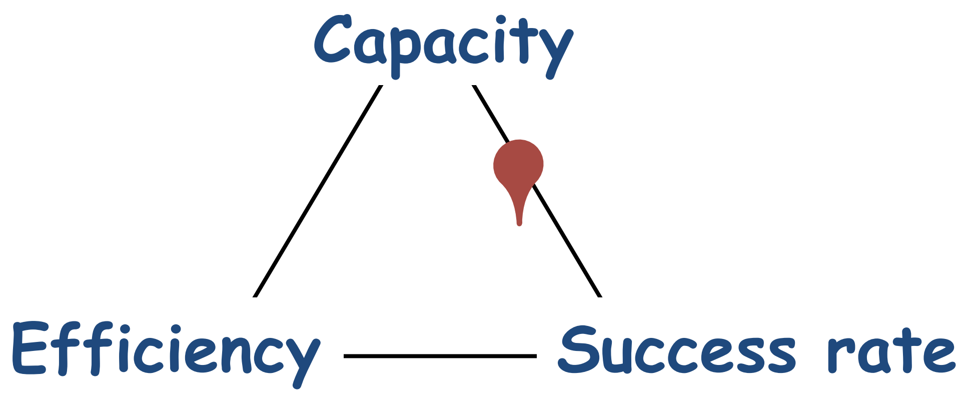
Capacity region of two-write WOM codes



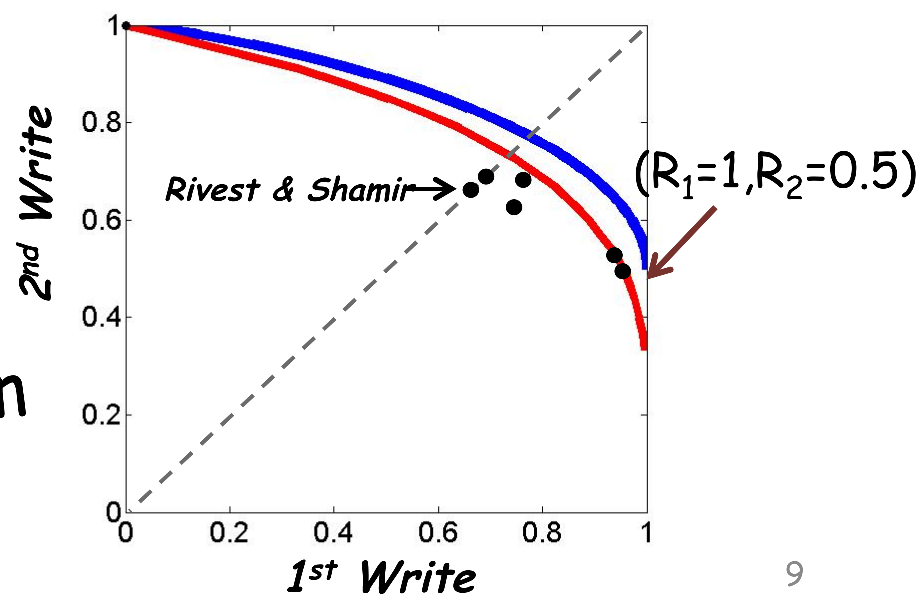
Yadgar, Yaakobi, Schuster, *Write once, get 50% free: Saving SSD erase costs using WOM code*, FAST '15

Our Approach

- Do not touch:
  - Interface
  - Complexity
  - Logical capacity

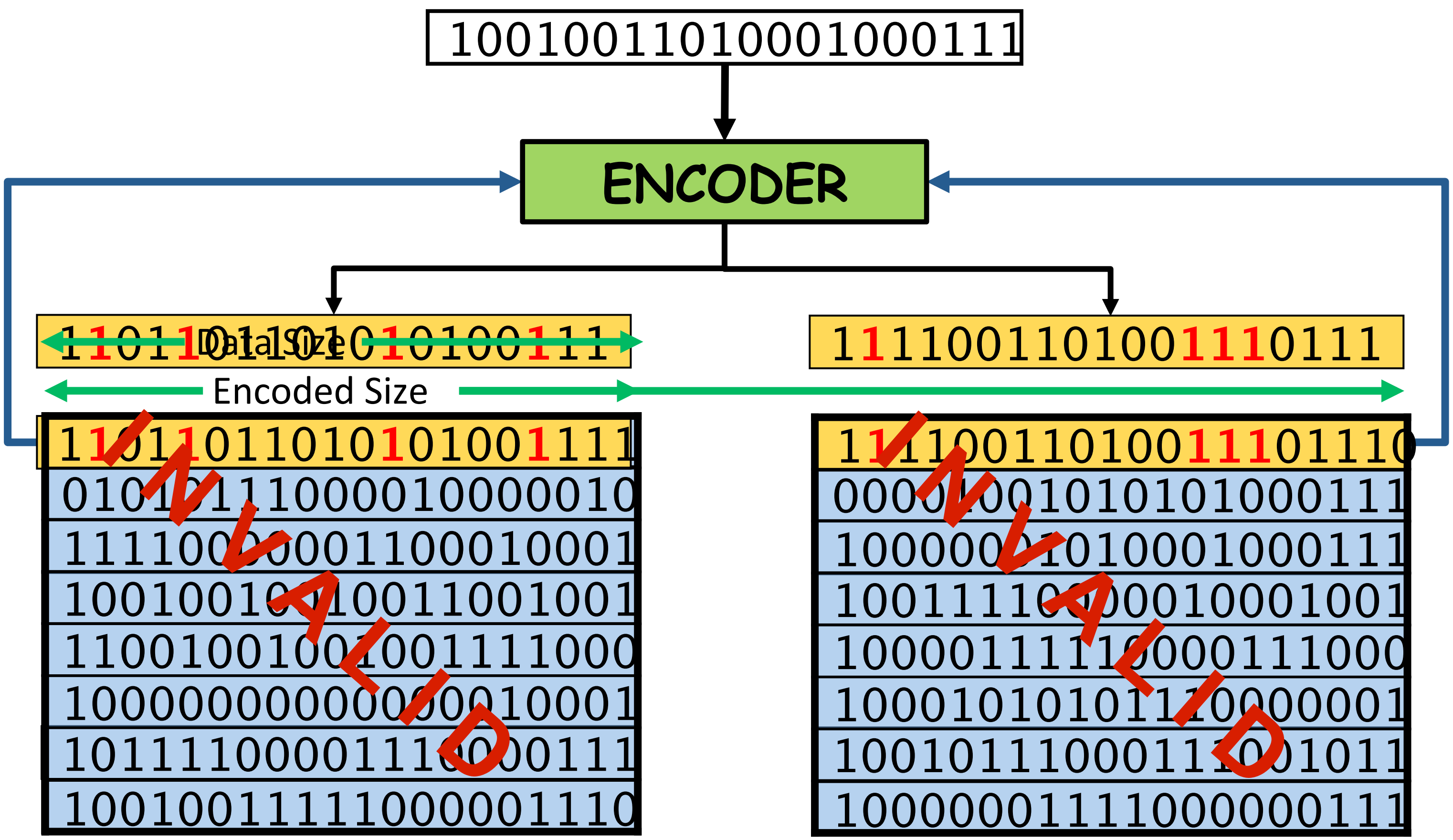


- Design handles:
  - Failures → retry
  - Latency → parallelism



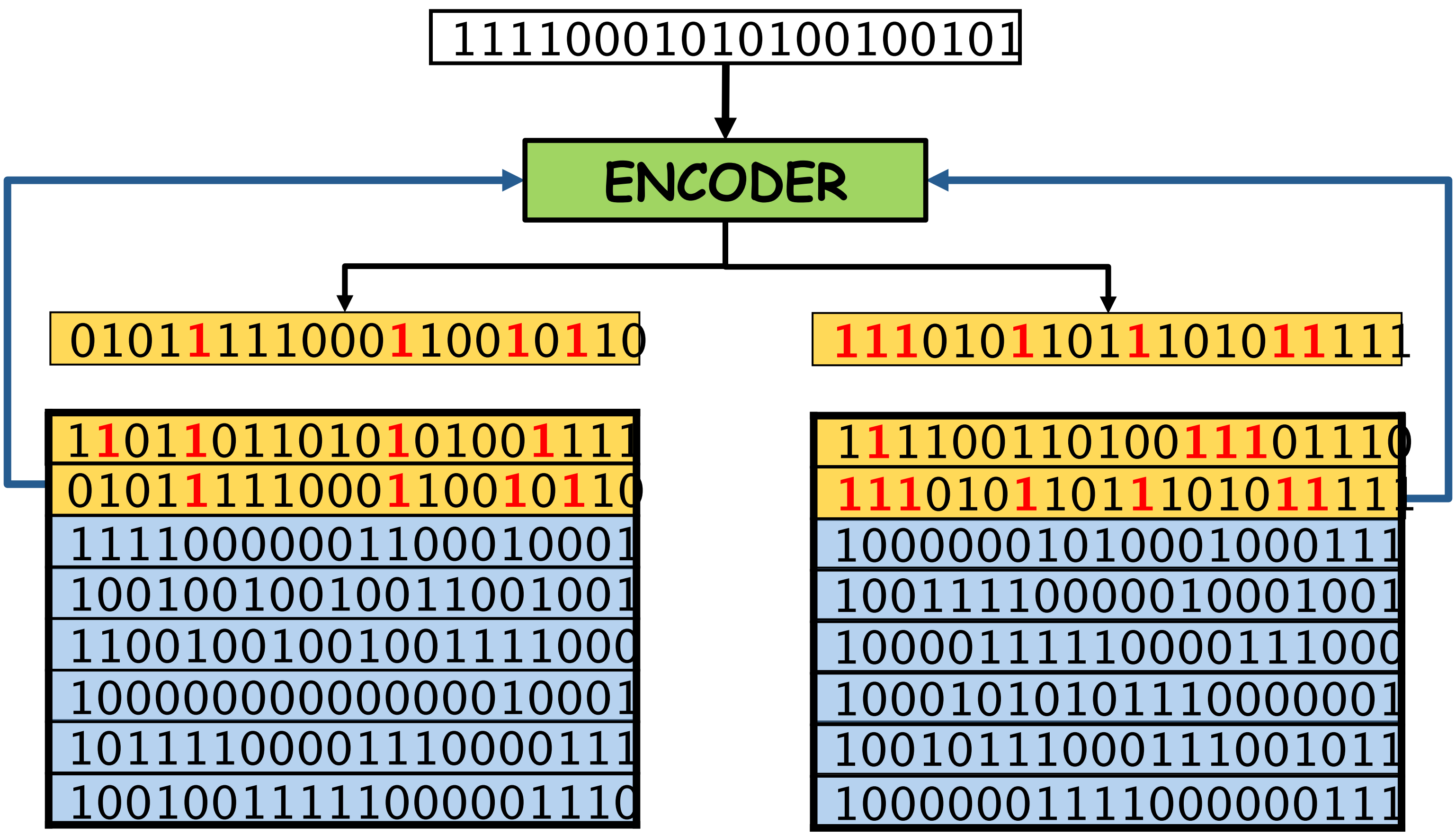
Reusable SSD

- **1st write**: (almost) unmodified → no overhead
- **2nd write**: one logical page → two physical pages



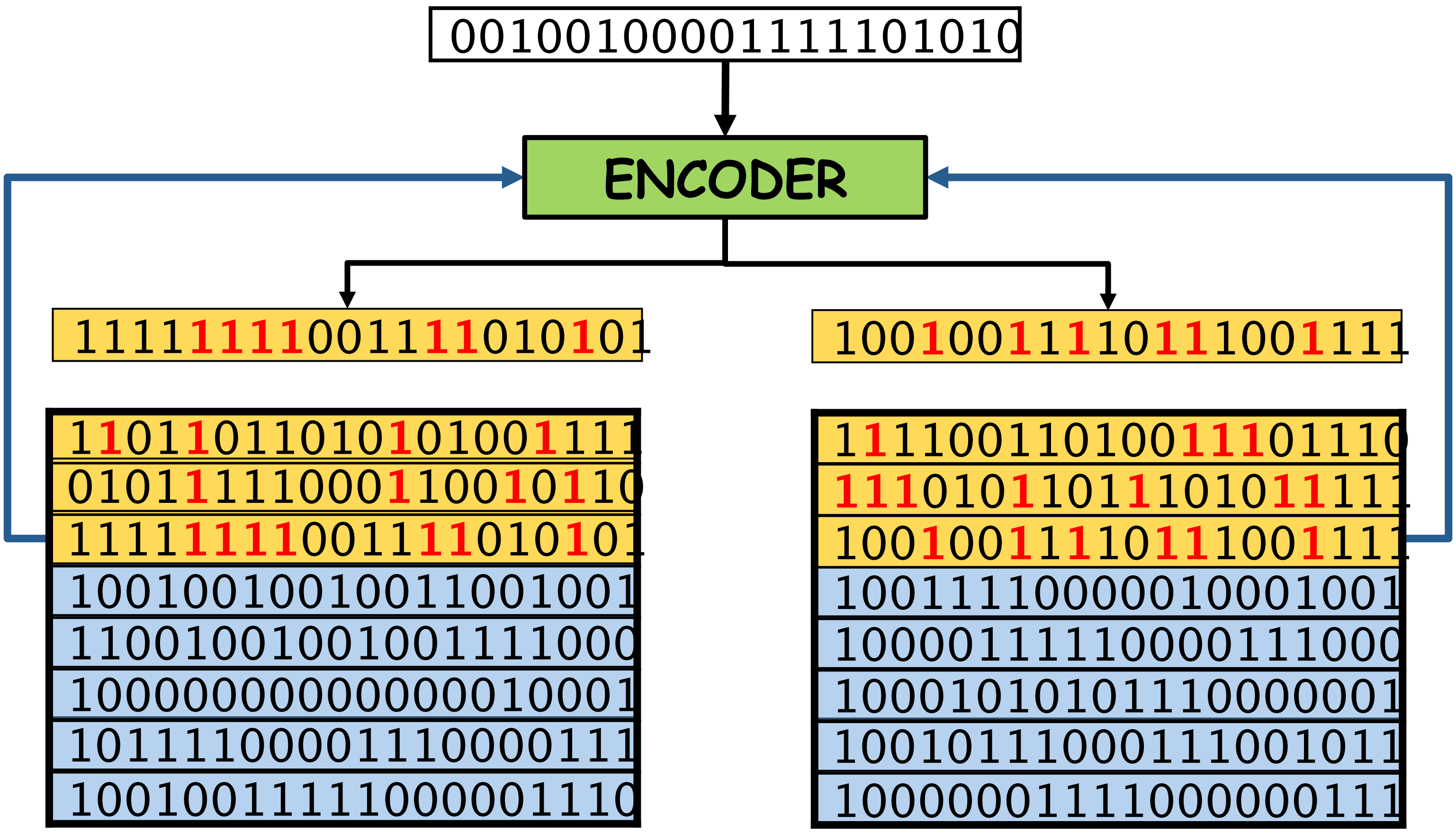
Reusable SSD

- **1st write**: (almost) unmodified → no overhead
- **2nd write**: one logical page → two physical pages



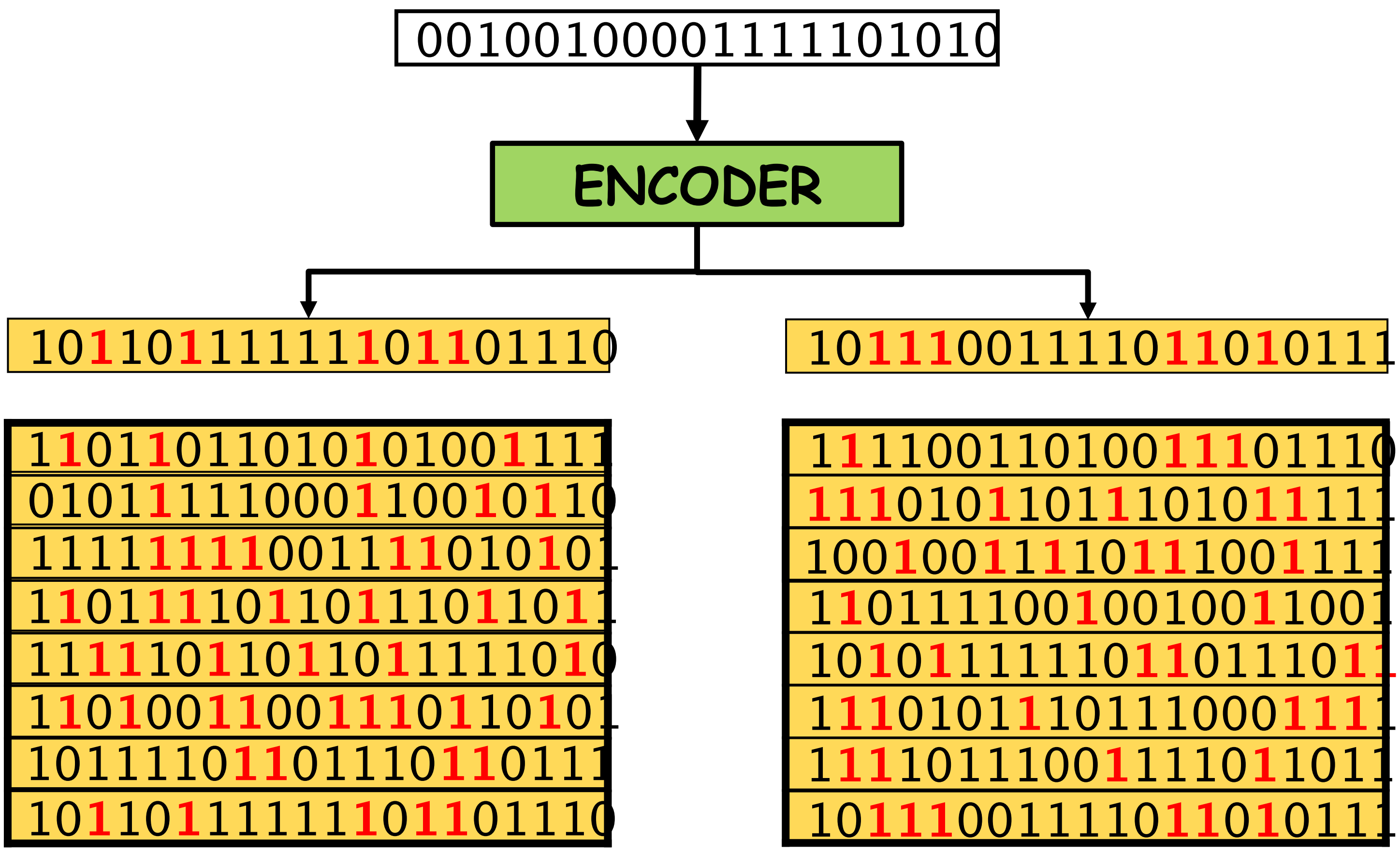
Reusable SSD

- **1st write**: (almost) unmodified → no overhead
- **2nd write**: one logical page → two physical pages



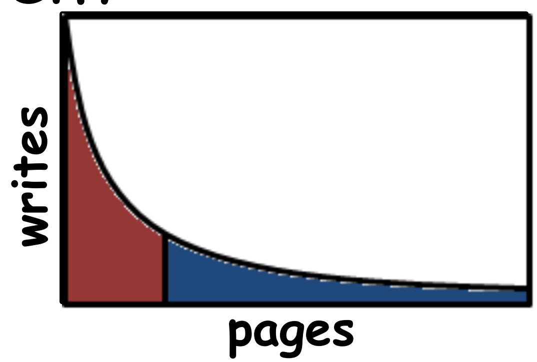
Reusable SSD

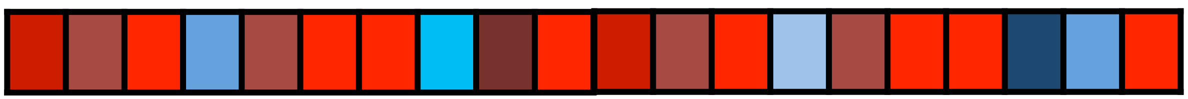
- **1st write**: (almost) unmodified → no overhead
- **2nd write**: one logical page → two physical pages



Hot/Cold Data

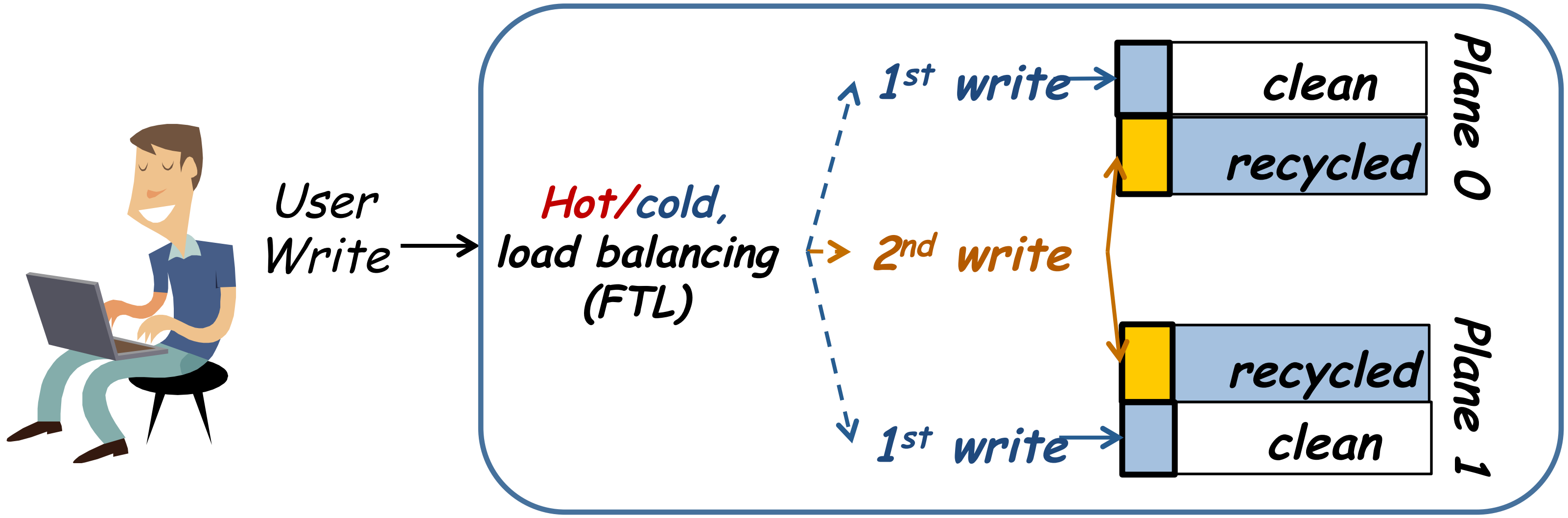
- First writes are more space efficient
  →Best for long term storage
  - *Hot data*: will be overwritten soon
  - *Cold data*: will remain valid for long

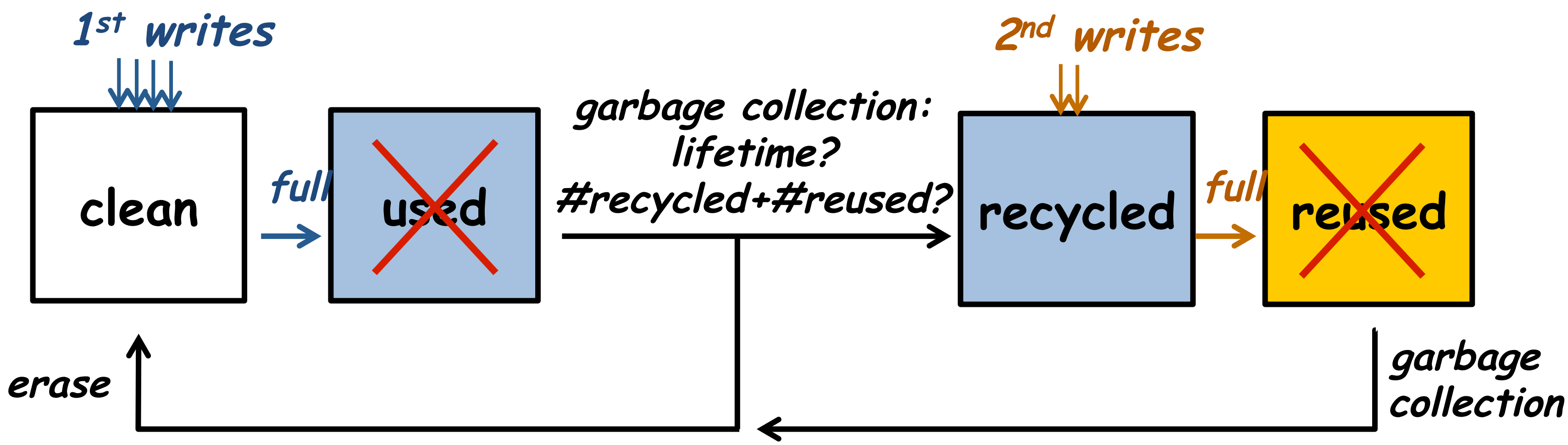




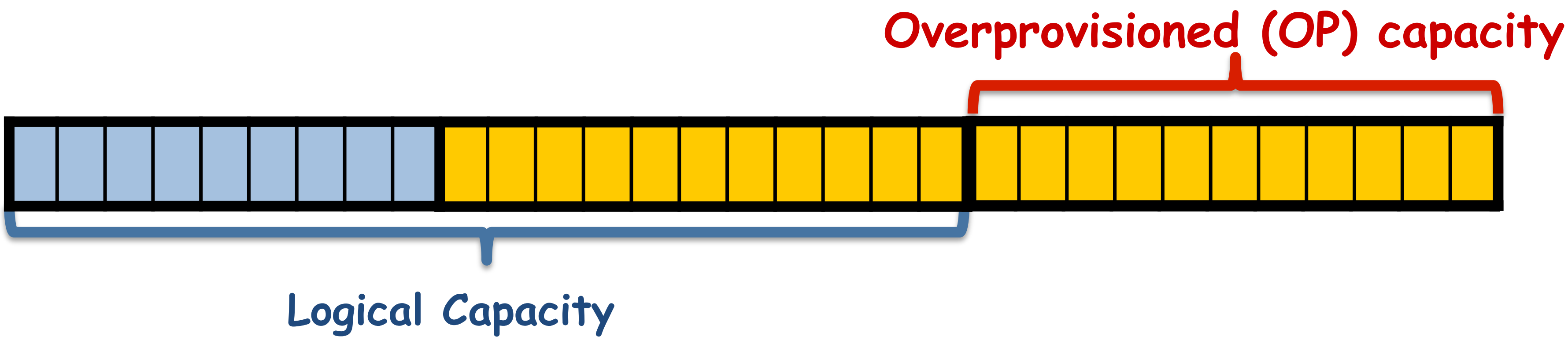
- Use second writes for hot pages
- Identify hot data according to I/O size
  - Heuristic : **small → hot**, **large → cold**
  - More accurate classifications available

Putting it All Together





Analysis



Erasures *Logical pages written*

- Standard SSD (best case): $E = N/Z$

Pages per block

- Reusable SSD (best case): "write once, get 50% free"

$$E' = N/(Z+Z/2) = 2/3E$$

→ **33% reduction in erasures** (without GC)

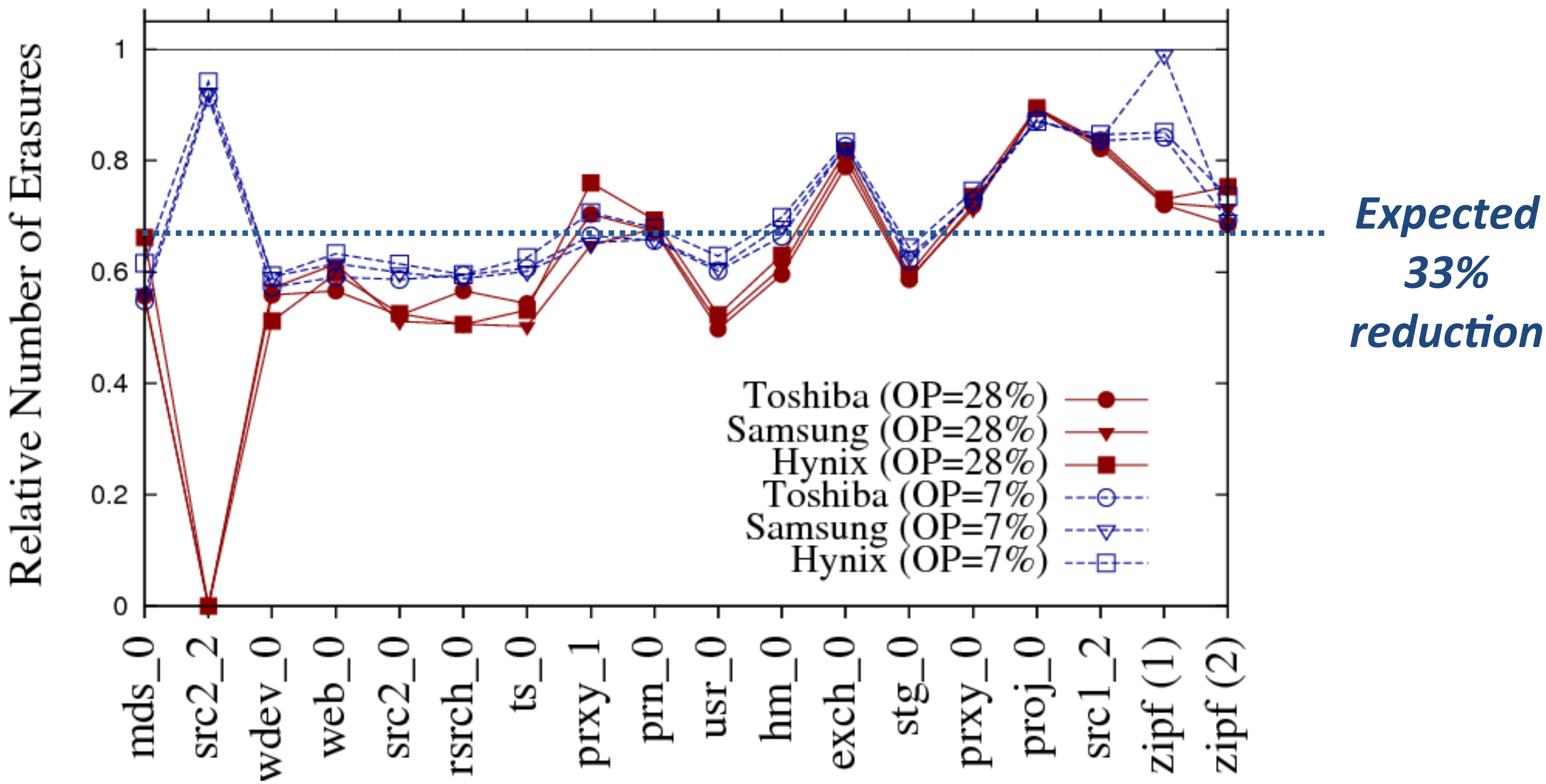
Evaluation

- How many **erasures** saved?
- How is **performance** affected?
- **Sensitivity** to design parameters

	Type	Pgs/Blk	R (us)	W (ms)	E (ms)
TOSHIBA	SLC	64	30	0.3	3
SAMSUNG	MLC	128	200	1.3	1.5
hynix	MLC	256	80	1.5	5

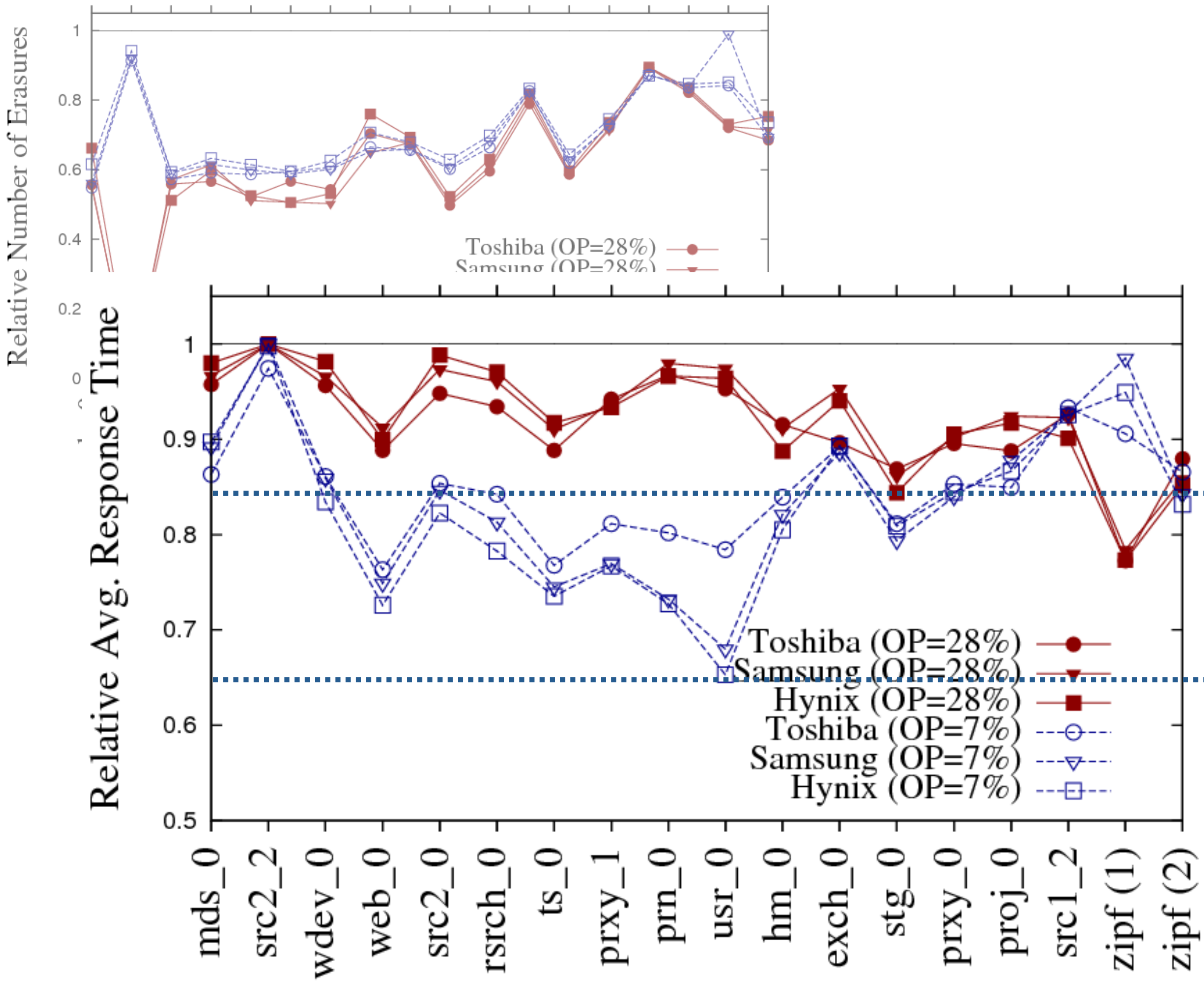
- DiskSim simulator
  - Available SSD extension
  - Modified FTL component
- Trace input:
  - Microsoft MSR + Exchange
  - Synthetic Zipf

Erasures



Relative Number of Erasures
1
0.8
0.6
0.4
0.2
0
Toshiba (OP=28%)
Samsung (OP=28%)
Hynix (OP=28%)
Toshiba (OP=7%)
Samsung (OP=7%)
Hynix (OP=7%)
mds_0
src2_2
wdev_0
web_0
src2_0
rsrch_0
ts_0
prxy_1
prn_0
usr_0
hm_0
exch_0
stg_0
prxy_0
proj_0
src1_2
zipf (1)
zipf (2)
Expected
33%
reduction

Response Time

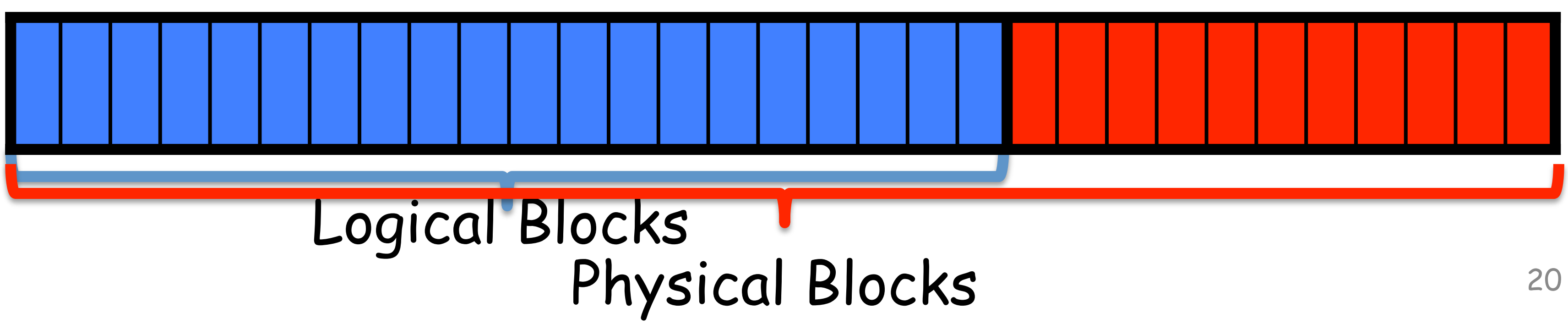


Enterprise: up to 15% reduction

Consumer: up to 35% reduction

Greedy Garbage 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?
- **Theorem** [Hu & Haas '10]: Greedy garbage collection is optimal in order to reduce the write amplification (for uniform writing)



Greedy Garbage Analysis

- **Write amplification** = $\dfrac{\text{\# Physical writes}}{\text{\# Logical writes}}$
- **Overprovisioning** = (T-U)/U ;

  T

Erasure Factor = 1/(1-α')

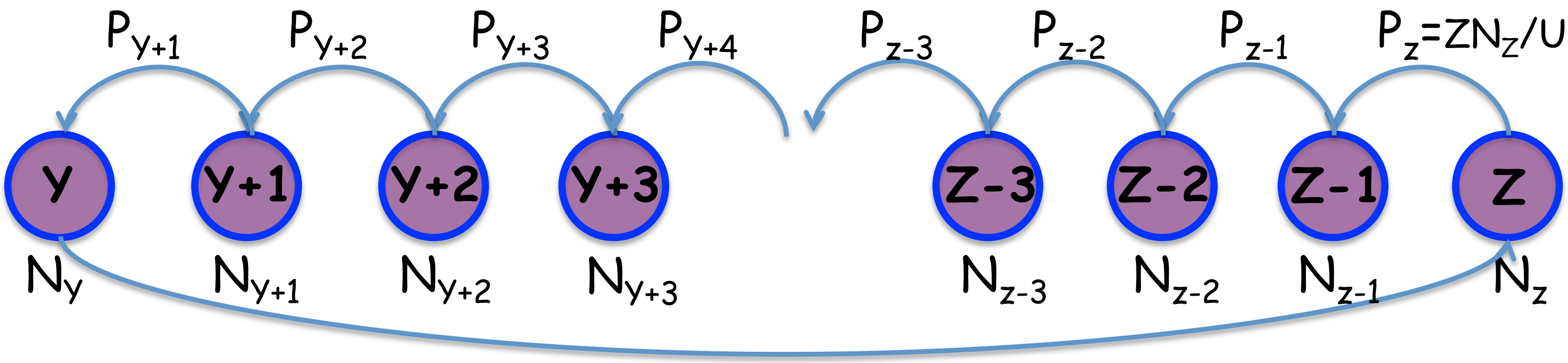
- **Quest** ... nd write amplification related, ***under random uniform writing***?
  - **N** = #logical page writes ; **M** = # physical page writes
  - **E** = #block erasures = M/Z , **Z** = # pages in a block
- On average: **Y = α'Z** valid pages in an erased block
  - **M = N + EY**
  - E = M/Z = (N+EY)/Z; (Z-Y)E = N; E=N/(Z-Y);

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

- **Overprovisioning** = (T-U)/U ;
  T = #physical pages, U = #logical pages
- **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

Erasure Factor = 1/(1-α')

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



This process forms a Markov chain which will be in a **steady state**

N_i = # of blocks with i valid logical pages

P_i = the prob for a block to move from state i to state i-1 = iN_i/U

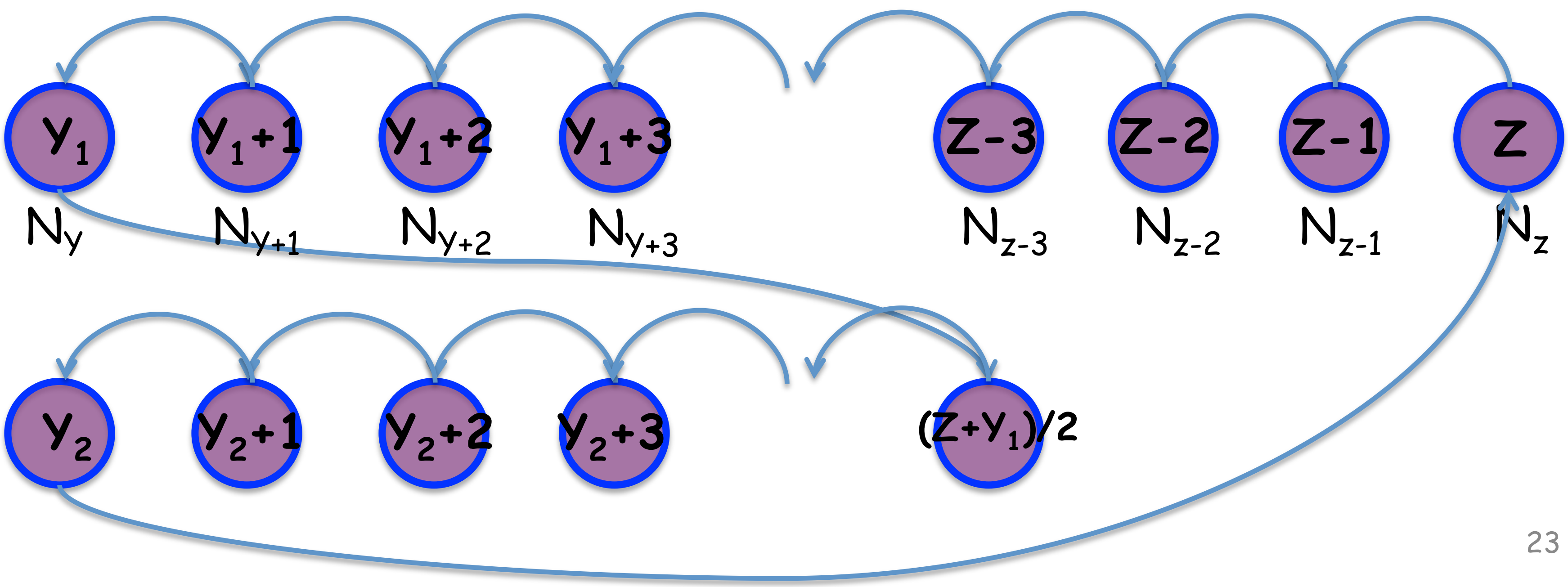
Key property: The number of **pages** in every state is (roughly) fixed

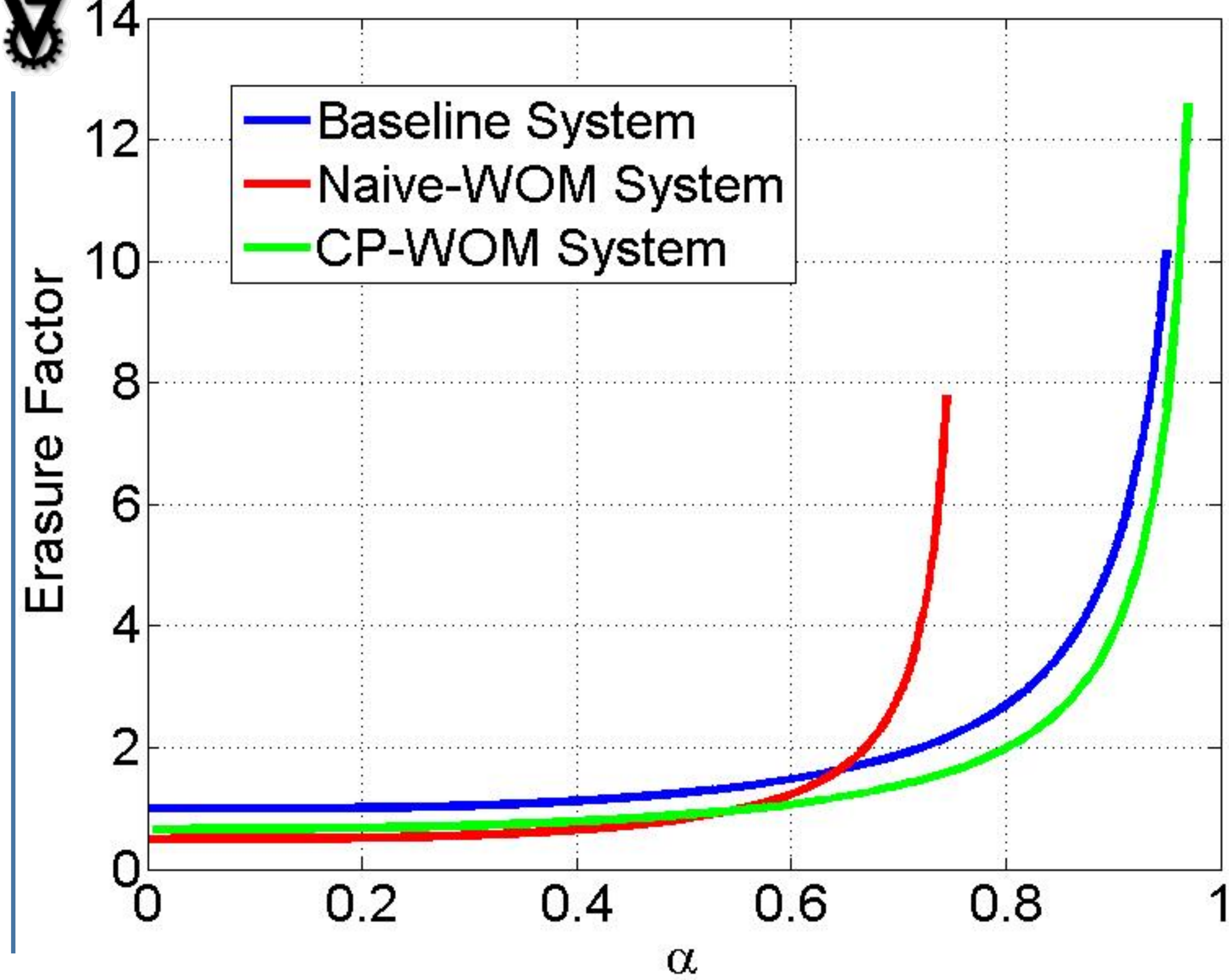
$$i \bullet N_i = \text{constant}$$

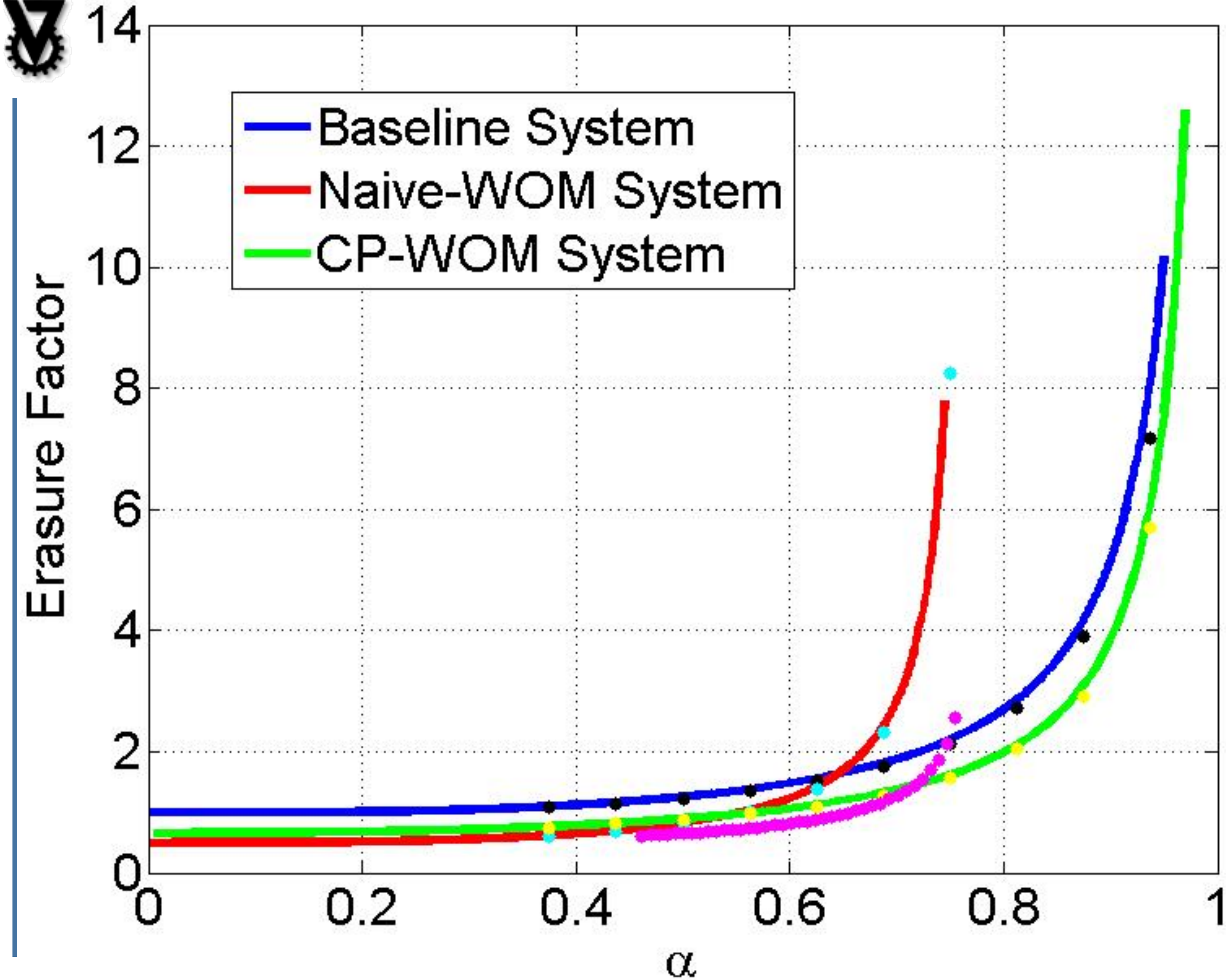
α = U/T= ··· ≈ (Z-Y)/(Zln(Z/Y)) = (Y/Z-1)/ln(Z/Y) = **(α'-1)/ln(α')**

Analysis of Reusable SSD

- Blocks can be in 2 different states: first or second write
- **GC algorithm**: characterized by a parameter γ_1 –
  - B_1, B_2: the blocks with min number of valid pages on a 1st, 2nd write
  - If the number of valid pages in B_1 is at most $Y_1=\gamma_1 \cdot Z$, move the block to 2nd write
  - Otherwise, erase the block B_2 and copy its valid pages




Baseline System
Naive-WOM System
CP-WOM System
Erasure Factor
α
0
2
4
6
8
10
12
14
0
0.2
0.4
0.6
0.8
1



Baseline System
Naive-WOM System
CP-WOM System
Erasure Factor
α
0
2
4
6
8
10
12
14
0
0.2
0.4
0.6
0.8
1

Summary

- An **applicable** design of re-writes in SSD
  - Unmodified logical capacity
  - No hardware modification
  - Efficient encoding
- Erasures analysis
  - Up to **50% additional free writes**
  - Studying the improvement in erasure factor via WOM codes
  - Extensions for multiple writes
- More to be done
  - Improved codes
  - Multiple writes
  - Analysis of different workloads
  - Understanding the rewriting effect on the memory lifetime

Thank you!