



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

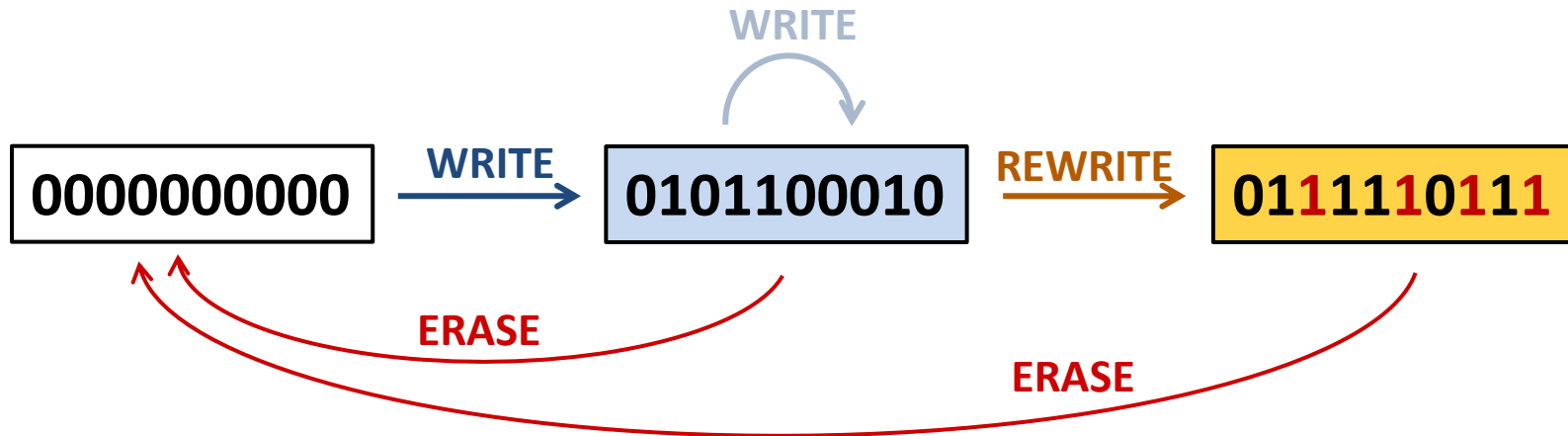
Gala Yadgar, Eitan Yaakobi, and Assaf Schuster

Computer Science

Technion



Reusing Flash with WOM Codes

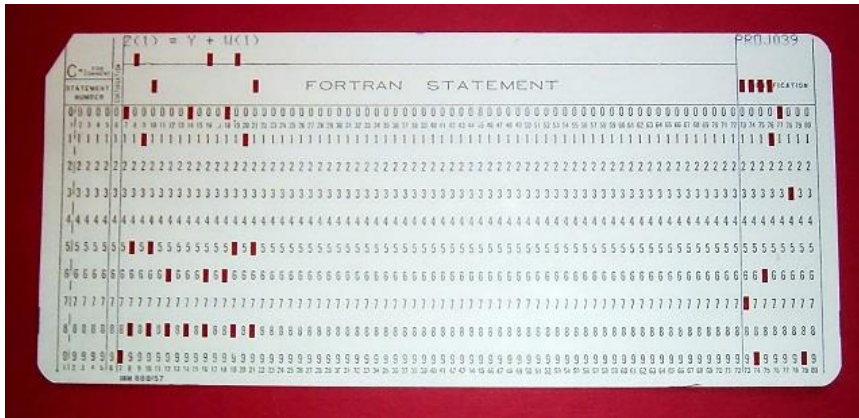


Reusable SSD:

- Perform additional writes before erasure
- Extend **lifetime**
- Improve **performance**
- Preserve **capacity** and **interface**



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)

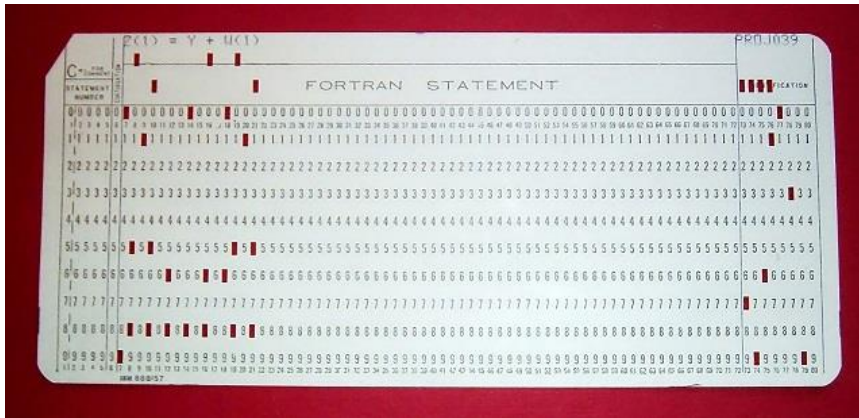
- $0 \rightarrow 1$ but $1 \nrightarrow 0$
- WOM Code: write n bits of information on m cells, $n > m$
- Example: write **11** and then write **01**

– Normally: 

– WOM code: 



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)

- $0 \rightarrow 1$ but $1 \nrightarrow 0$
- WOM Code: write n bits of information on m cells, $n > m$
- Example: write **11** and then write **01**

– Normally: 

– WOM code: 



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

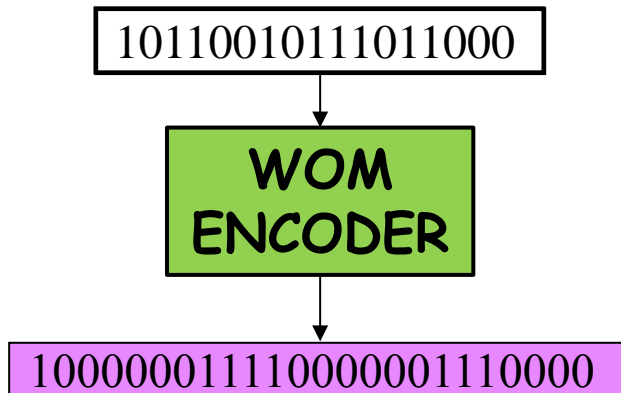
| data | 1 st write | 2 nd write |
|------|-----------------------|-----------------------|
| 00 | 000 | 111 |
| 10 | 100 | 011 |
| 01 | 010 | 101 |
| 11 | 001 | 110 |

← Data Size →

← Encoded Size →

| |
|--------------------------|
| 101100110100000011100010 |
| 00001010101010001110000 |
| 100000010100010001110000 |
| 100111100001100010011000 |
| 100001111100001110001000 |
| 100010101011100000010000 |
| 100101110001110010110000 |
| 100000011110000001110000 |

[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]

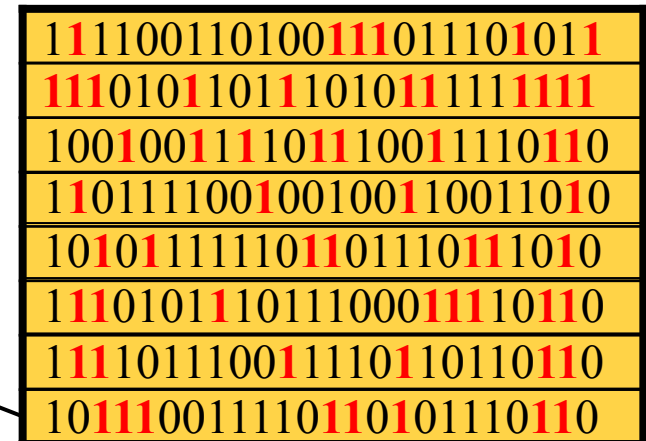
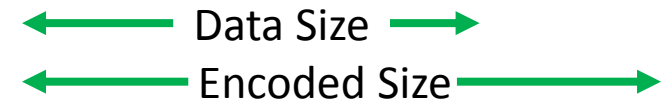




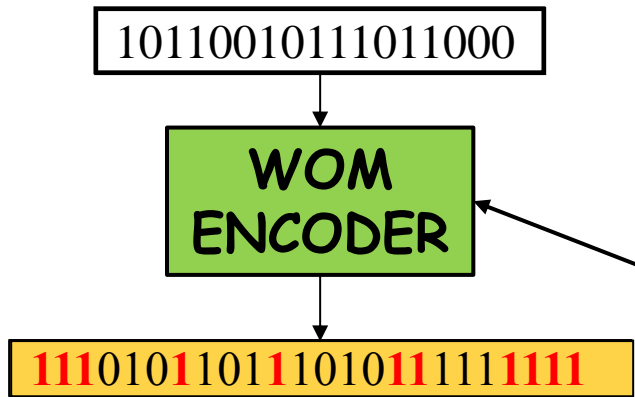
Typical Use of WOM Codes

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

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



[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]

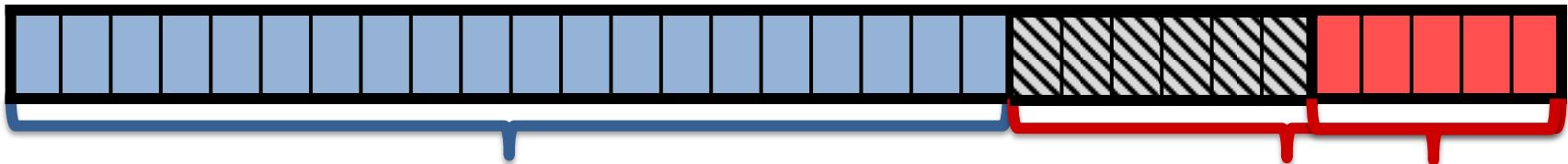




Drawbacks of Typical Use

- Capacity overhead:

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



- Performance overheads:

- I/O operations access 29%-50% **more bits**
- A **read precedes** every second write

- Compatibility:

- **Requires modification** in physical page size

- Applicability:

- High encoding/decoding **complexity**



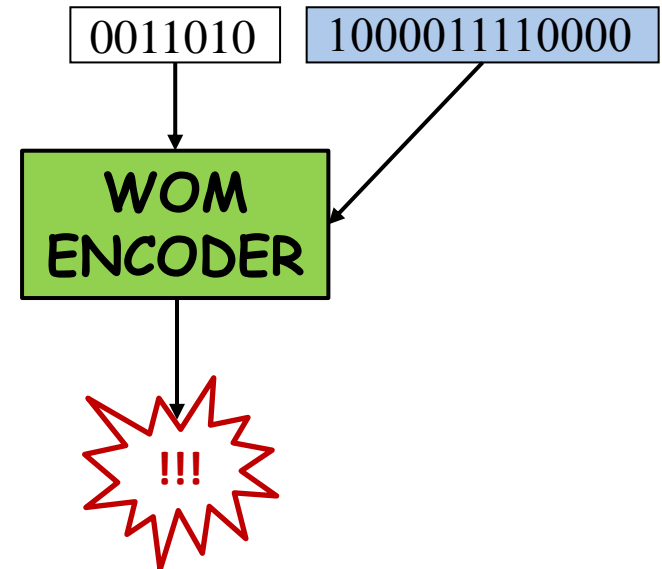
Reusable SSD Design Goals

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

Polar WOM Codes

[Burshtein and Strugatski, IT '13]

- **Efficient**
- **May fail...**





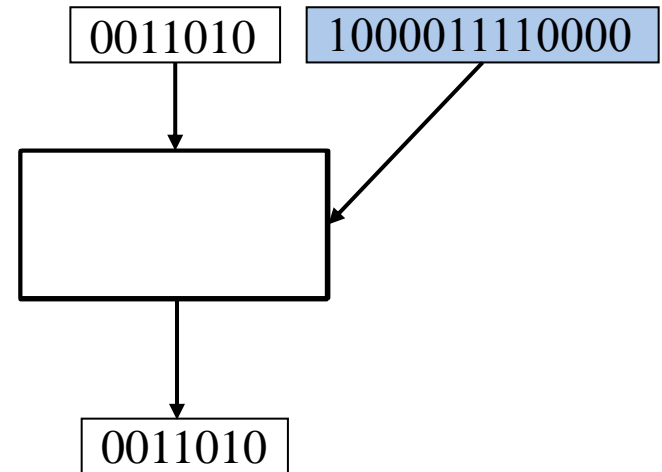
Reusable SSD Design Goals

- Do not touch:
 - Interface
 - Performance
 - Logical capacity
- Avoid **latency** by **parallelizing**
- Handle **failures** by **retrying**
 - 5% of encodings fail
 - 0.25% of encodings fail twice
 - Fall back to 1st write

Polar WOM Codes

[Burshtein and Strugatski, IT `13]

- **Efficient**
- **May fail...**





Reusable SSD

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

100100110100010001110000



← Data Size →

← Encoded Size →

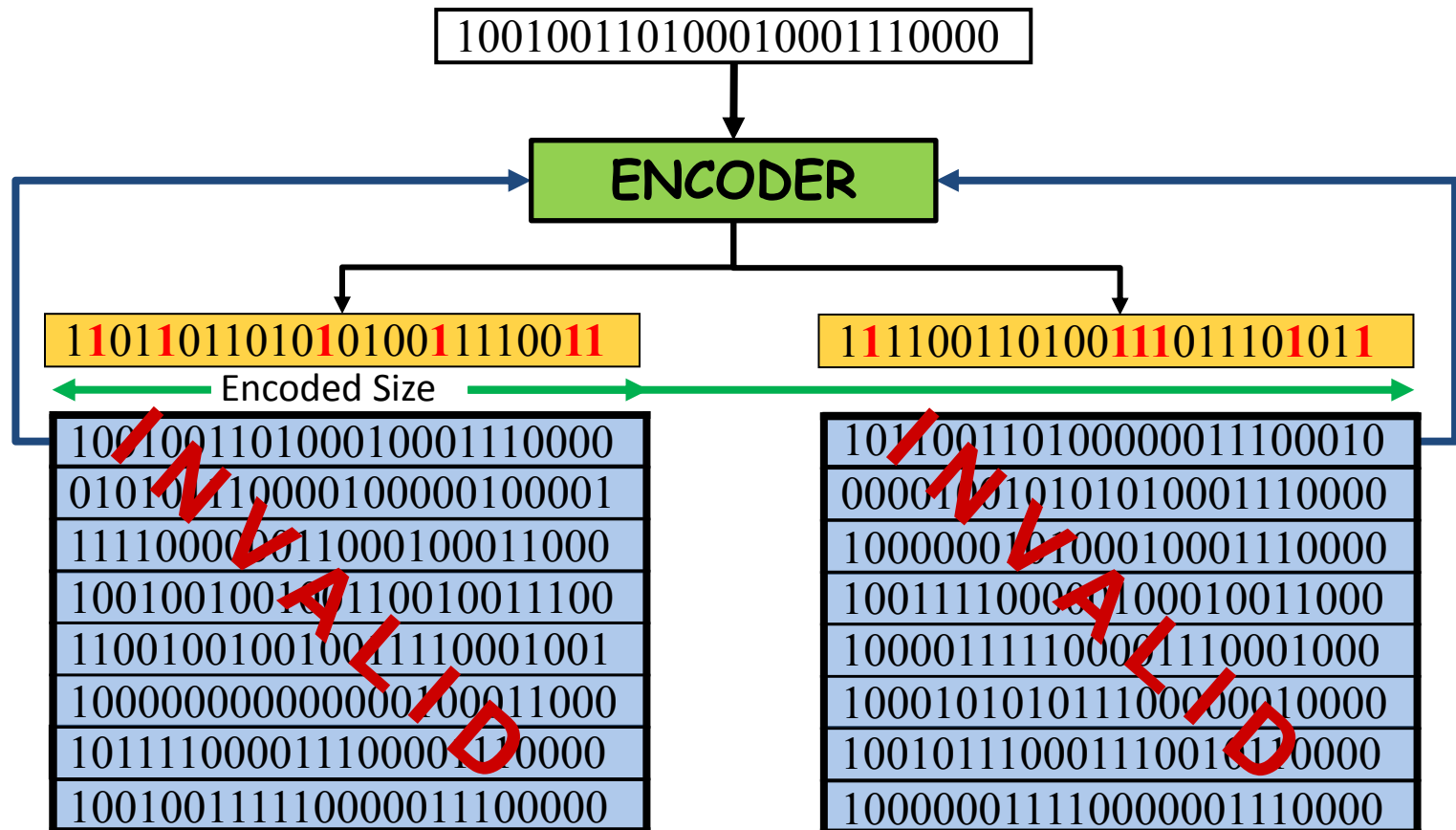
| |
|---------------------------|
| 100100110100010001110000 |
| 010101110000100000100001 |
| 111100000911000100011000 |
| 100100100101110010011100 |
| 110010010010011110001001 |
| 1000000000000000100011000 |
| 10111100001110000110000 |
| 100100111110000011100000 |

| |
|--------------------------|
| 101100110100000011100010 |
| 00001010101010001110000 |
| 10000001000010001110000 |
| 10011110000100010011000 |
| 100001111100001110001000 |
| 100010101011100000010000 |
| 100101110001110010110000 |
| 100000011110000001110000 |



Reusable SSD

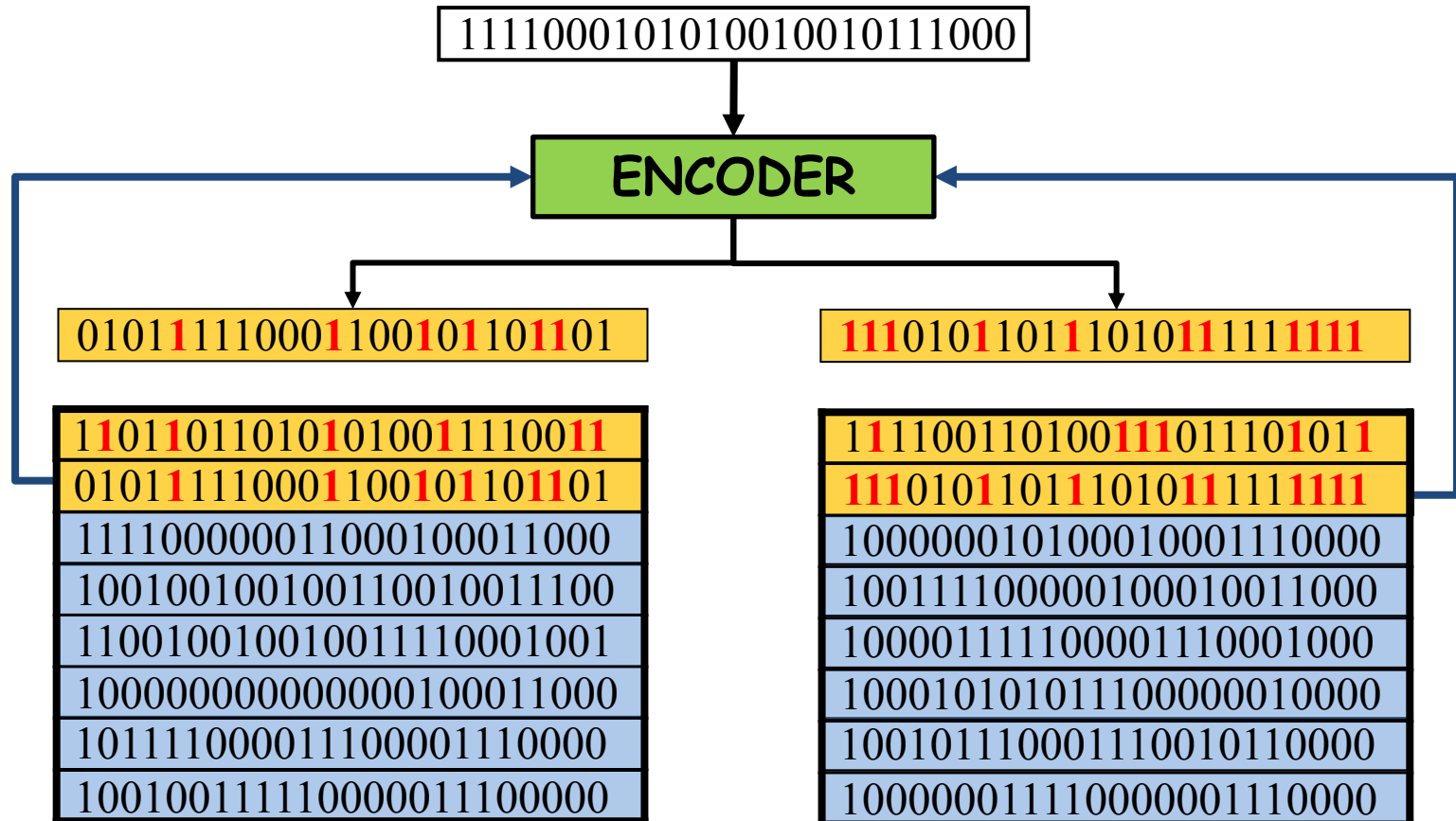
- **1st write:** (almost) unmodified → no overhead
- **2nd write:** one logical page → two physical pages





Reusable SSD

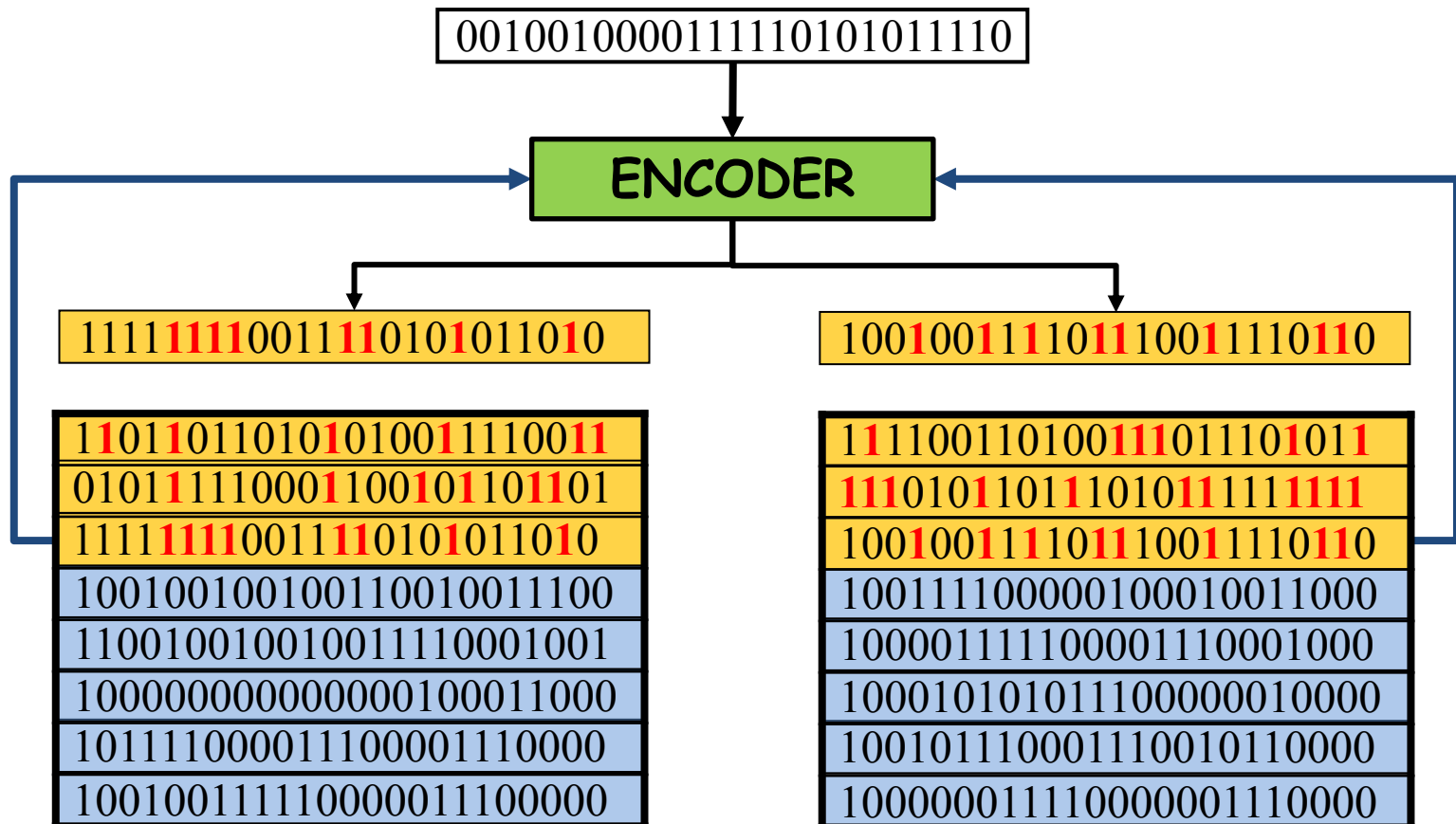
- **1st write:** (almost) unmodified → no overhead
- **2nd write:** one logical page → two physical pages





Reusable SSD

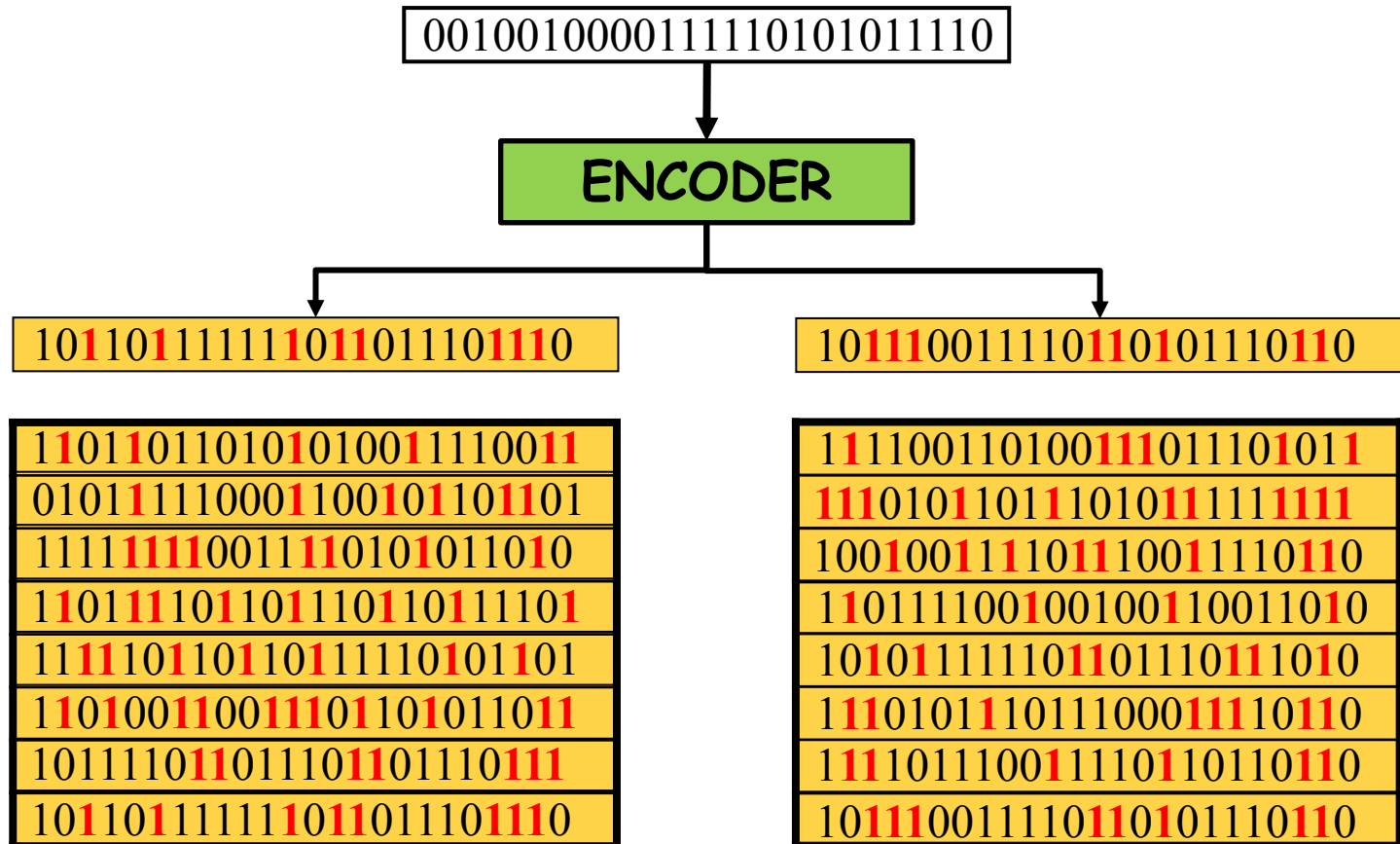
- **1st write:** (almost) unmodified → no overhead
- **2nd write:** one logical page → two physical pages





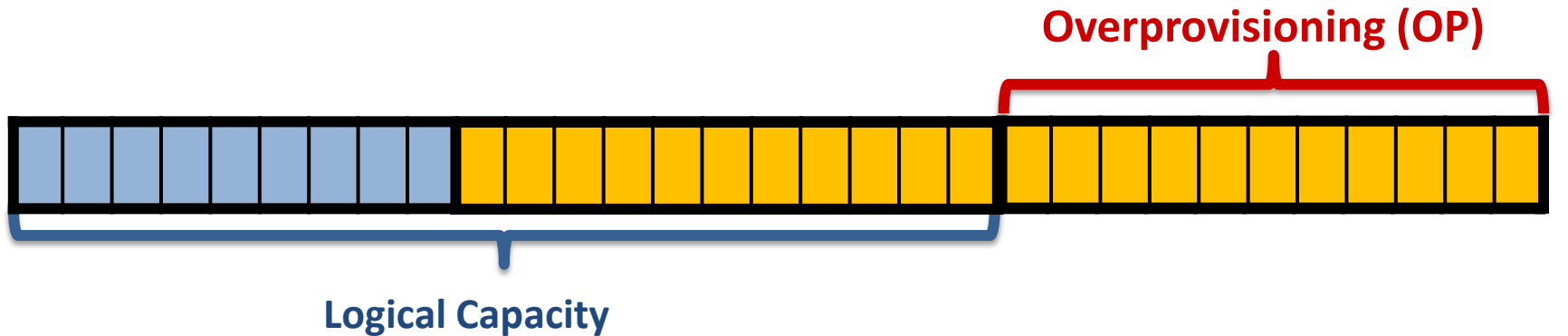
Reusable SSD

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





The Benefit of Reusable SSD



- Standard SSD (best case):
$$E = \frac{M}{N}$$

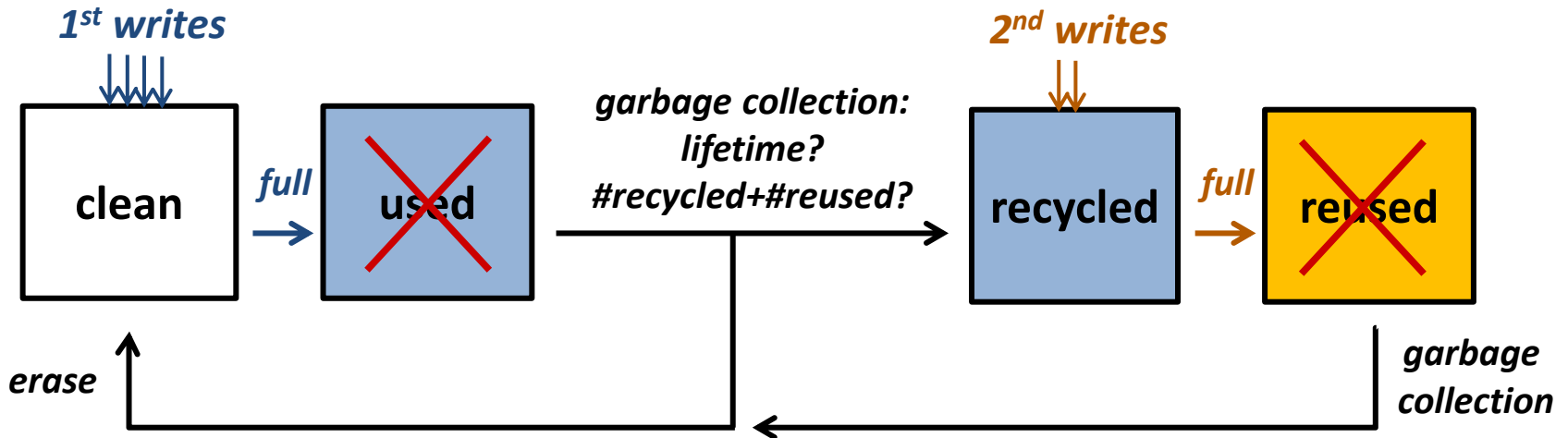
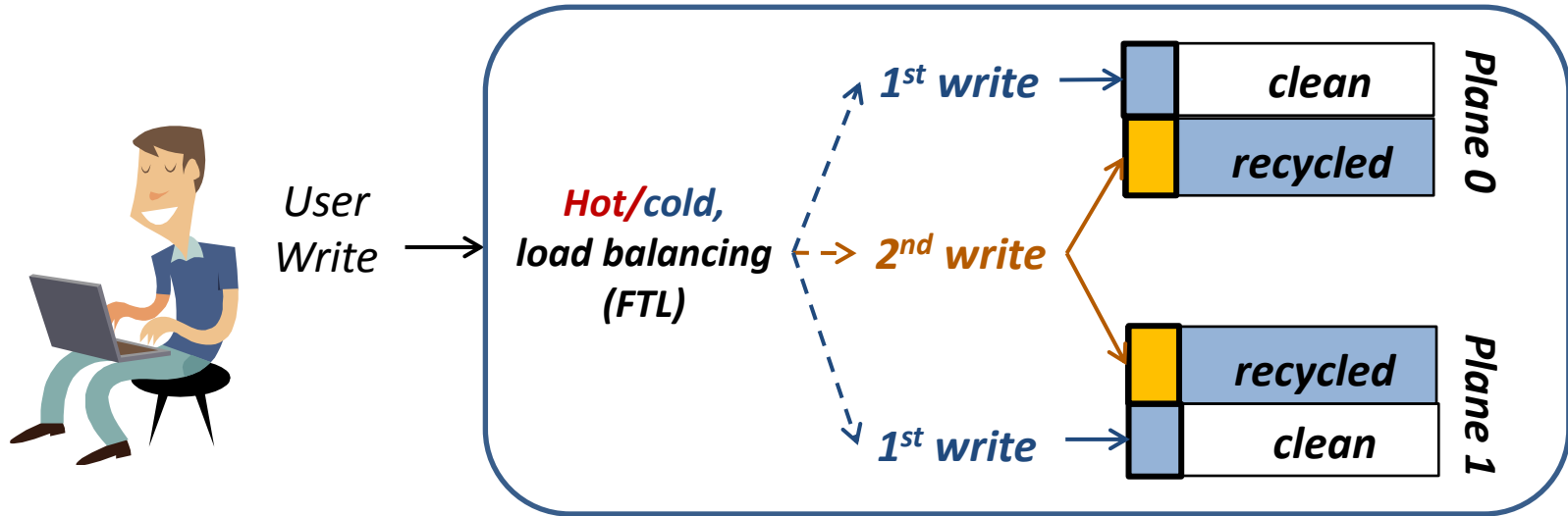
E — Erasures
M — Logical pages written
N — Pages per block
- Reusable SSD (best case): “write once, get 50% free”

$$E' = \frac{M}{N + N/2} = \frac{2}{3} E$$

→ **33% reduction in erasures**



Putting it All Together





Extending SSD Lifetime

- User level
 - Caching, admission control, specialized FS/DB...
[Soundararajan et. al., FAST '10][Oh et. al., FAST '12]
[Li et. al., ATC '14][Marmol et. al., HotStorage '14]
- FTL
 - Wear leveling, throttling, partitioning, buffering, deduplication...
[Agarwal et. al., ATC '08][Lee et. al., FAST '12]
[Desnoyers TOS '14][Kim and Ahn, FAST '08]
[Gupta et. al., FAST '11]
- Code
 - Error correction
[Cai et. al., DATE '12][Pan et. al., FAST '11][Zhao et. al., FAST '13]
- Flash
 - Write voltage/speed [Jeong et. al., FAST '14]



SSD

FTL



...110101011101...

Flash





Evaluation

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

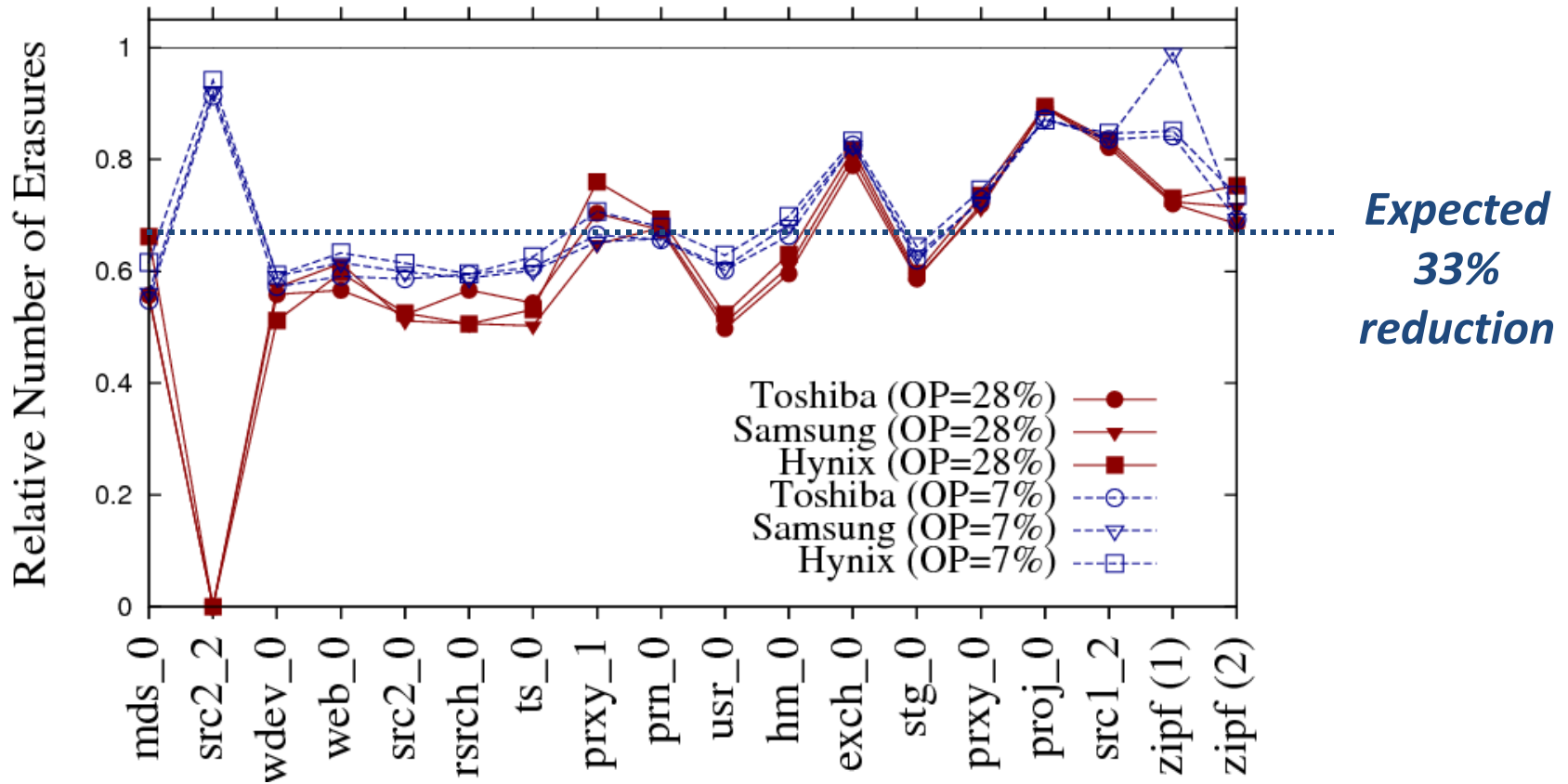


| Type | Pgs/Blk | R (us) | W (ms) | E (ms) |
|------|---------|--------|--------|--------|
| SLC | 64 | 30 | 0.3 | 3 |
| MLC | 128 | 200 | 1.3 | 1.5 |
| MLC | 256 | 80 | 1.5 | 5 |

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

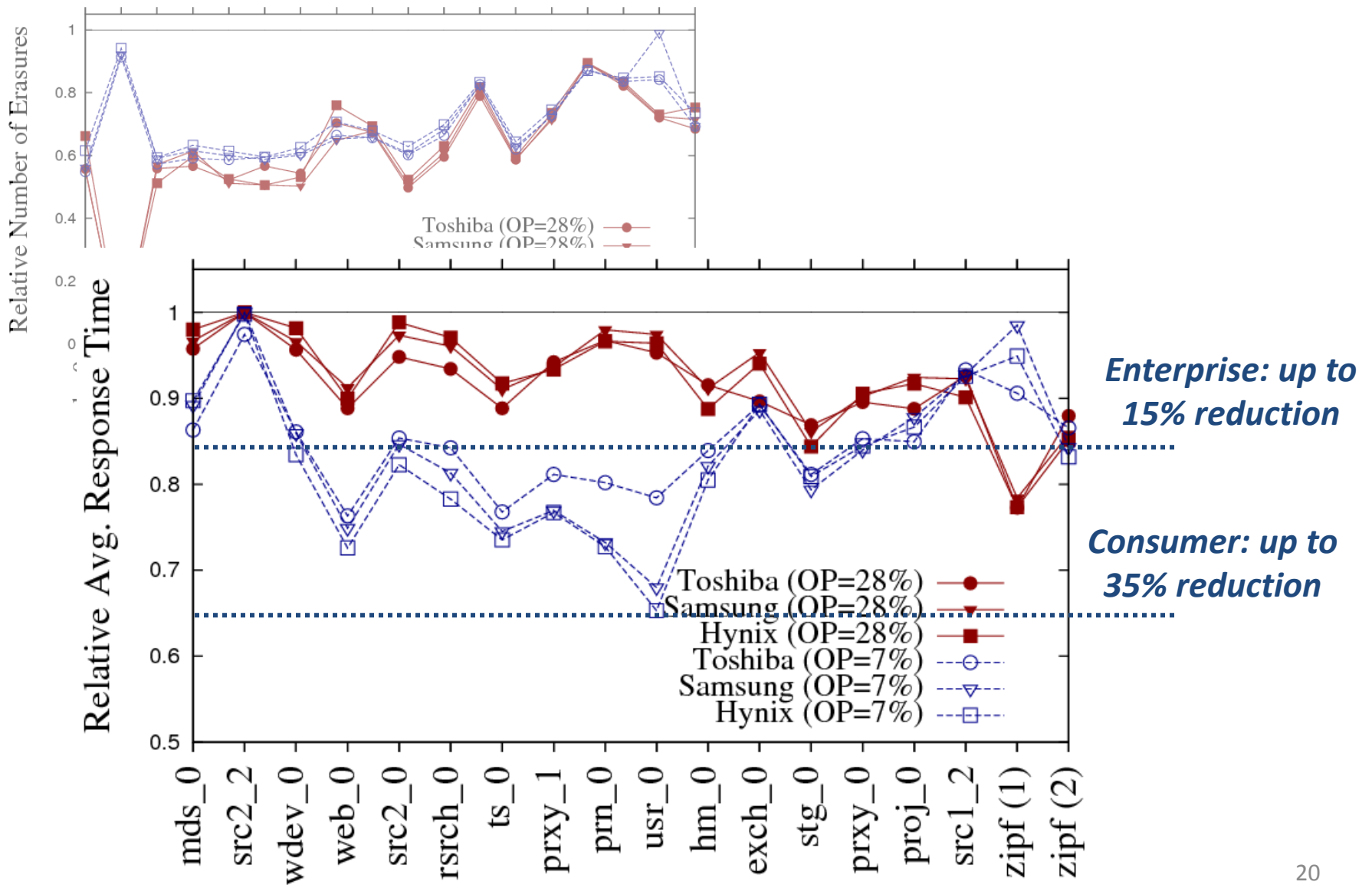


Erasures





Response Time





Summary

- First **applicable** design of re-writes in SSD
 - ✓ Unmodified logical capacity
 - ✓ No hardware modification
 - ✓ Efficient computation
- Up to **50% additional free writes**
- Less erasures → improved **performance**
 - ✓ Parallel 2nd writes
 - ✓ Prefetching
 - ✓ Independent retries

Thank you!

- **This is only the beginning!**
 - Improved codes
 - Hardware encoders