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 Translation Layer (FTL)**

- **Logical Capacity**: physical space reserved for logical space
- **Physical Capacity**: actual physical storage that can be written to
- **Overprovisioned (OP) Capacity**: space reserved in physical storage to avoid wear and tear on the physical space
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

• **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

(Rivest and Shamir, 1982)
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

<table>
<thead>
<tr>
<th>data</th>
<th>1st write</th>
<th>2nd write</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>000</td>
<td>111</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>011</td>
</tr>
<tr>
<td>01</td>
<td>010</td>
<td>101</td>
</tr>
<tr>
<td>11</td>
<td>001</td>
<td>110</td>
</tr>
</tbody>
</table>

[Grupp et. al., MICRO ’09][Jagmohan et.al., MSST ’10]
[Louie 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
  – 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

(R₁=1, R₂=0.5)
Our Approach

• Do not touch:
  – Interface
  – Complexity
  – Logical capacity

• Design handles:
  – Failures → retry
  – Latency → parallelism

Capacity

Efficiency  —  Success rate

\( R_1 = 1, R_2 = 0.5 \)
Reusable SSD

- **1\textsuperscript{st} write**: (almost) unmodified $\rightarrow$ no overhead
- **2\textsuperscript{nd} write**: one logical page $\rightarrow$ two physical pages

\[
\begin{array}{c}
10010011010001000111
\end{array}
\]

\[
\begin{array}{c}
1111001101001110111
\end{array}
\]

\[
\begin{array}{c}
11011011010100111
\end{array}
\]

\[
\begin{array}{c}
10010011010001000111
\end{array}
\]

\[
\begin{array}{c}
1111001101001110111
\end{array}
\]

\[
\begin{array}{c}
10010011010001000111
\end{array}
\]

\[
\begin{array}{c}
1111001101001110111
\end{array}
\]

\[
\begin{array}{c}
10010011010001000111
\end{array}
\]

\[
\begin{array}{c}
1111001101001110111
\end{array}
\]

\[
\begin{array}{c}
10010011010001000111
\end{array}
\]
Reusable SSD

- **1\textsuperscript{st} write**: (almost) unmodified $\rightarrow$ no overhead
- **2\textsuperscript{nd} write**: one logical page $\rightarrow$ two physical pages
Reusable SSD

- **1st write**: (almost) unmodified $\rightarrow$ no overhead
- **2nd write**: one logical page $\rightarrow$ two physical pages
Reusable SSD

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

![Diagram with binary numbers and encoder process]
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

User Write

Hot/cold, load balancing (FTL)

1st write

2nd write

1st writes

2nd writes

clean

recycled

recycled

clean

full

garbage collection: lifetime? #recycled+#reused?

full

garbage collection

used

erase

reused
Analysis

- Standard SSD (best case): $E = \frac{N}{Z}$
- Reusable SSD (best case): “write once, get 50% free”

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

→ 33% reduction in erasures (without GC)
Evaluation

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Pgs/Blk</th>
<th>R (us)</th>
<th>W (ms)</th>
<th>E (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLC</td>
<td>64</td>
<td>30</td>
<td>0.3</td>
<td>3</td>
</tr>
<tr>
<td>MLC</td>
<td>128</td>
<td>200</td>
<td>1.3</td>
<td>1.5</td>
</tr>
<tr>
<td>MLC</td>
<td>256</td>
<td>80</td>
<td>1.5</td>
<td>5</td>
</tr>
</tbody>
</table>

• **DiskSim** simulator
  – Available SSD extension
  – Modified FTL component

• **Trace input:**
  – Microsoft MSR + Exchange
  – Synthetic Zipf
Erasures

Relative Number of Erasures

Toshiba (OP=28%)  Toshiba (OP=7%)
Samsung (OP=28%)  Samsung (OP=7%)
Hynix (OP=28%)    Hynix (OP=7%)

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** = \( \frac{(T-U)}{U} \);
  \[ T = \# \text{physical blocks}, \; U = \# \text{logical blocks} \]

- **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 = \( \frac{\text{# Physical writes}}{\text{# Logical writes}} \)

• Overprovisioning = \( \frac{T-U}{U} \) ;
  \( T = \# \text{physical pages}, U = \# \text{logical pages} \)

• Question: How are the overprovisioning factor and write amplification related, under random uniform writing?
  – \( N = \# \text{logical page writes} \); \( M = \# \text{physical page writes} \)
  – \( E = \# \text{block erasures} = M/Z \), \( Z = \# \text{pages in a block} \)

• On average: \( Y = \alpha'Z \) 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 is the connection b/w \( \alpha = U/T \) and \( \alpha' = Z/Y \)?
• Answer: \( \alpha = (\alpha' - 1)/\ln(\alpha') \) (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

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)

Erasure Factor = 1 / (1 - α')

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 • N_i = constant

α = U / T = ... ≈ (Z - Y) / (Z ln(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 $y_1$ -
  – $B_1$, $B_2$: the blocks with min number of valid pages on a 1\textsuperscript{st}, 2\textsuperscript{nd} write
  – If the number of valid pages in $B_1$ is at most $Y_1 = y_1 \cdot Z$, move the block to 2\textsuperscript{nd} write
  – Otherwise, erase the block $B_2$ and copy its valid pages
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