Reducing Solid-State Storage Device Write Stress Through Opportunistic In-Place Delta Compression

Xuebin Zhang, Jiangpeng Li, Hao Wang, Kai Zhao and Tong Zhang
xuebinzhang.rpi@gmail.com

ECSE Department, Rensselaer Polytechnic Institute (RPI)
Outline

- Introduction and Motivation
- Progressive Programmability of SLC Flash Pages
- In-place Delta Compression Realization in SSDs
- Evaluation Results and Overhead Analysis
- Future Work and Summary
NAND Flash Memory

- Increasing Adoptions and Decreasing Cost of NAND Flash Memory

- Endurance: One of Existing Problems of NAND Flash Memory
  - Make it worse: Bad management in software stack.
Content Temporal Locality in Storage Systems

- Small (<4kB) update will trigger a 4kB write in Flash (write amplification)
  - 60% writes are smaller than one page (4kB), even less than 10 bytes

  Redundancy between consecutive updates

- Frequently repeated in-place update within a short time period

Repeated visiting times

<table>
<thead>
<tr>
<th>Workload Traces</th>
<th>Finance-1</th>
<th>Finance-2</th>
<th>Homes</th>
<th>Webmail</th>
</tr>
</thead>
<tbody>
<tr>
<td>% of LBAs</td>
<td>1</td>
<td>2-10</td>
<td>11-100</td>
<td>&gt;1000</td>
</tr>
<tr>
<td></td>
<td>101-1000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

% of LBAs: 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Delta Compression: Problems in Current Practice

- Latency: need to read multiple pages to recover the latest version.
- Storage: need to keep a mapping table to store the delta’s location.
Outline

- Introduction and Motivation
- Progressive Programmability of SLC Flash Pages
- In-place Delta Compression Realization in SSDs
- Evaluation Results and Overhead Analysis
- Summary and Future Work
NAND Flash Progressive Programming

- Append new data in the same page (for SLC flash pages)
  - Writing ‘1’ to flash cell will not change its state
  - Overwrite new data to original physical page
NAND Flash Progressive Programming

- Hardware Platform: PCIe interface, FPGA as the Flash controller
Validity checking of multiple programming before erasing

- Conventional: one program – one erase
- Progressive: eight program – one erase
Outline

- Introduction and Motivation
- Progressive Programmability of SLC Flash Pages
- In-place Delta Compression Realization in SSDs
- Evaluation Results and Overhead Analysis
- Summary and Future Work
Proposed Solution: Combine Data/Delta Compression

- Use intra-sector lossless data compression to make space for deltas

- Per-sector Compression ratio distribution of different file types

![Compression ratio distribution graph]

Smaller is better
Proposed Solution: In-place Delta Compression

- Overall flow diagram to update one new version of sector data

  - Read from SLC-mode page
  - Data reconstruction
  - Delta compression
  - Header generation and ECC encoding
  - Allocate a new SLC-mode page
  - Compress $C_k$ and write to the allocated page
  - Enough space?
  - Write $p_k$ through partial programming

- Sector content change

  $$d_0 d_1 \cdots d_n 111 \cdots 111 \cdots 11$$

  OR

  $$d_0 d_1 \cdots d_n p_1 p_2 \cdots p_m 1 \cdots 11$$
Proposed Solution: In-place Delta Compression

- Consecutive in-place updates

<table>
<thead>
<tr>
<th>Physical page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
</tr>
<tr>
<td>$C_0$</td>
</tr>
<tr>
<td>$d_1$</td>
</tr>
<tr>
<td>$d_2$</td>
</tr>
<tr>
<td>$\vdots$</td>
</tr>
<tr>
<td>$d_k$</td>
</tr>
</tbody>
</table>

1st update

2nd update

Full: $k$-th update

Allocate a new page

Reset delta compression

<table>
<thead>
<tr>
<th>$P_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_0$</td>
</tr>
<tr>
<td>$d_1$</td>
</tr>
<tr>
<td>$d_2$</td>
</tr>
<tr>
<td>$\vdots$</td>
</tr>
<tr>
<td>$d_k$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{k+1}$</td>
</tr>
</tbody>
</table>
Proposed Solution: Data Placement

- Two different data placement strategies
  - Clustered: shared region for deltas of this physical page
  - Segmented: independent regions for deltas of each sector only

![Diagram showing clustered and segmented placement strategies]
Proposed Solution: ECC Management

- ECC management for different types of data elements
  - Compressed original sector data
  - Compressed deltas
  - Header for each elements

SLC-mode flash memory page

Compressed 4kB sector
Compressed delta
Header
ECC redundancy

One ECC codeword
Outline

- Introduction and Motivation
- Progressive Programmability of SLC Flash Pages
- In-place Delta Compression Realization in SSDs
- Evaluation Results and Overhead Analysis
- Summary and Future Work
Case study: file system metadata

- Use a benchmark to generate database/files operations
- Implement a metaAnalyzer to grasp metadata from file system
- Analyze the collected consecutive versions of metadata

Delta compression efficiency: (full metadata size is 256 bytes)
Evaluation: Write Stress Reduction

- Write stress reduction to store 1000 consecutive versions

Less than 50
Evaluation: Write Stress Reduction

- General cases analysis
  - Set the data/delta compression ratio as Gaussian distribution
  - Simulation driven by traces of different workloads

![Graph showing normalized page used count for different workloads: Webmail, Repeated File Update, Homes, TPC-C. Segmented and Clustered cases. Lower is better.]

(a) Webmail
Evaluation: Write Stress Reduction

- General cases analysis

![Chart showing normalized page used count for different cases]

Lower is better.
Evaluation: Read Latency Overhead

- Read operation’s flow diagram in proposed solution

- Read latency model
  - Conventional Practice without data/compression:
    \[ \tau_{\text{read}} = \tau_{\text{sen}} + \tau_{\text{xfer}} (4kB) + \tau_{\text{LDPC}}^{(\text{dec})} + \tau_{\text{sata}} \]
  - Proposed solution:
    \[ \tau_{\text{read}} = \tau_{\text{sen}} + \tau_{\text{xfer}} (n \cdot 4kB) + \max(\tau_{\text{LDPC}}^{(\text{dec})}, \tau_{\text{BCH}}^{(\text{dec})}) + \max(\tau_{\text{sec}}, \tau_{\text{delta}}) + \tau_{\text{com}} + \tau_{\text{sata}} \]
## Evaluation: Latency Overhead and Silicon Cost

- **Flash memory parameter configurations**
  - Flash memory sensing latency: 40 us
  - Data transfer from flash to controller (ONFI 4.0): 800MB/s
  - LDPC/BCH decoding throughput: 1GB/s

- **Read latency overhead**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Technique</th>
<th>Average-case (us)</th>
<th>Worst-case (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Conventional</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clustered</td>
<td>76</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>Segmented</td>
<td>56</td>
<td>63</td>
</tr>
</tbody>
</table>

- **Silicon cost**
  - Total involved silicon area is **0.39 mm²** at 22nm, while in general, a whole controller’s silicon area size is around **10-20 mm²**.
Outline

- Introduction and Motivation
- Progressive Programmability of SLC Flash Pages
- In-place Delta Compression Realization in SSDs
- Evaluation Results and Overhead Analysis
- Summary and Future Work
Extensions and Future Work

- Use a hardware accelerator to offload computation intensive tasks
  - Will not bring troubles to controller because of small silicon size and power consumption
  - Benefit a lot from the perspective of performance

- Exploit the byte-addressability to realize the delta compression in NVM
  - Limited endurance, expensive cost
  - Inherent support of byte-addressability
Conclusion

- SLC Flash page can support “Progressive Partial Programming”:
  different portions of the same flash page can be programmed at different time.

- Lossless data compression can be utilized to make space for deltas between consecutive versions of updates.

- SSD write stress can be reduced by up to 80% with proposed solution without significant overhead.

Thank you!