Deduplication in SSDs: Model and Quantitative Analysis

Jonghwa Kim, Sangyup Lee, Ikjoon Son, Jongmoo Choi, Dankook University, Korea
ChoongHyun Lee, Massachusetts Institute of Technology
Sungroh Yoon, Korea University, Korea
Hu-ung Lee, Sooyong Kang, Youjip Won, Jaehyuk Cha, Hanyang University, Korea
Contents

- Introduction
- Deduplication Overview
- Architecture for Deduplication on SSD
- Cost-benefit Model
- Three Acceleration Techniques
  - SHA-1 hardware logic (H/W approach)
  - Sampling based Filtering (S/W approach)
  - Recency based Fingerprint Management (S/W approach)
- Experimental Environments
- Experimental Results
- Conclusion
Introduction

- Deduplication
  - Remove duplicate writes
  - Reduce storage utilization
  - Improve I/O performance

- Additional Benefits of Deduplication in SSD
  - Improve garbage collection overhead (especially highly utilized cases)
  - Enhance lifetime by reducing WAF (Write Amplification Factor)
  - Eliminate mapping overhead for deduplication using FTL

- Issues
  - Deduplication Overhead
    - How much is the deduplication overhead in SSD, especially considering the limited resources of SSD?
  - Duplication Rate
    - Are there enough duplicate data for obtaining performance gain in SSD workloads?
SSD overview

- Main Components of SSD
  - SATA host interface
  - SSD controller
  - An array of flash chips.
- Characteristics of Flash Memory
  - Erase-before-write
  - Limited number of program/erase cycles.
- FTL (Flash Translation Layer)
  - Out-of-place update and wear-leveling mechanism
Internals of Deduplication Layer

- **Fingerprint Generator**
  - Fixed chucking vs. Variable chucking ➔ Fixed chucking (4KB)
  - Diverse cryptographic hash functions ➔ SHA-1
    - generate a 160-bit hash value from 4KB data
- **Fingerprint Manager**
  - # of fingerprints: Full vs. Partial ➔ Partial (Recent X fingerprints with LRU)
- **Mapping Manager**
  - Make use of the mapping table of FTL (page-level mapping)
Architecture for Deduplication

Interactions among the Fingerprint generator, Fingerprint manager and Mapping manager.

1) A write request: [10, A]
   10: logical block number
   A: content

2) Create Fingerprint

3) Lookup: Duplicated Or Not

4) Flash programming

5) FTL Map update

6) Insert A' and Page number

[10, A] -> \( A' = \text{SHA-1}(A) \) -> Fingerprint Generator

"Interactions among the Fingerprint generator, Fingerprint manager and Mapping manager."
Architecture for Deduplication

1) A write request: [11,B]
11: logical block number
B: content

2) Create Fingerprint

3) Lookup: Duplicated Or Not

4) Flash programming

5) FTL Map update

6) Insert B' and Page number

$B' = \text{SHA-1}(B)$

[10,A] [11,B]

Write Request

Fingerprint Generator

Fingerprint Manager

Mapping Table

Flash Memory

10 100 103
11 100 103
12 100 A

$100$ $103$ $B$
Architecture for Deduplication

1) A write request: [12, A]  
   12: logical block number  
   A: content

2) Create Fingerprint

3) Lookup: Duplicated Or Not

4) FTL MAP update without Flash programming

A' = SHA-1(A)

[10,A] [11,B] [12,A]
Deduplication Model

- **Question**
  - How much duplication rate is required to obtain performance gain in SSD?

- **Cost-benefit Analysis**
  - Write latency
    - Without deduplication
      \[
      \text{Write latency} = \text{FM}_{\text{program}} + \text{MAP}_{\text{manage}}
      \]
    - With deduplication
      \[
      \text{Write latency} = \left( \text{FP}_{\text{generator}} + \text{FP}_{\text{manage}} + \text{MAP}_{\text{manage}} \right) \times \text{Duprate} + \left( \text{FP}_{\text{generator}} + \text{FP}_{\text{manage}} + \text{MAP}_{\text{manage}} + \text{FM}_{\text{program}} \right) \times (1 - \text{Duprate})
      \]

**Terminology**

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write\text{_latency}</td>
<td>Write latency (elapsed time to handle a write request)</td>
</tr>
<tr>
<td>MAP\text{_manage}</td>
<td>Mapping table update time</td>
</tr>
<tr>
<td>FP\text{_manage}</td>
<td>Lookup time in the Fingerprint manager</td>
</tr>
<tr>
<td>FM\text{_program}</td>
<td>Programming time on Flash memory</td>
</tr>
<tr>
<td>FP\text{_generator}</td>
<td>Fingerprint generation time</td>
</tr>
<tr>
<td>Dup\text{_rate}</td>
<td>Duplication Rate (Ratio between the duplicate data and total written data)</td>
</tr>
</tbody>
</table>
To yield the performance gain

- Equation 2) < Equation 1)

\[
\left( FP_{\text{generator}} + FP_{\text{manage}} + MAP_{\text{manage}} \right) \times Dup_{\text{rate}} + \left( FP_{\text{generator}} + FP_{\text{manage}} + MAP_{\text{manage}} + FM_{\text{program}} \right) \times (1 - Dup_{\text{rate}})
\]

\[
< FM_{\text{program}} + MAP_{\text{manage}}
\]

\[
Dup_{\text{rate}} > \frac{FP_{\text{generator}} + FP_{\text{manage}}}{FM_{\text{program}}}
\]
Deduplication Model

Graph notation
- X-axis: Deduplication overhead (FPgenerator + FPmanage), Y-axis: Duplication rate.
- Each line: Flash memory programming time (200, 800, 1300 and 2500 us, respectively).
- Area above the line: where we can obtain performance gain using deduplication.

Implication
- The minimum duplication rate decreases as the deduplication overhead decreases or as the program time becomes longer.
- Workloads with the higher duplication rate can yield larger deduplication benefit.
Lesson from the deduplication model

- Higher duplication rate, longer programming time, and lower deduplication overhead can give positive impacts on performance
- The duplication rate is determined by the characteristics of workloads
- The programming time is given by the specification of Flash memory
- The key point is how to reduce the deduplication overhead (FPgenerator, FPmanage, and MAPmanage)

Three Acceleration Techniques

- SHA-1 Hardware logic (H/W approach)
- Sampling based Filtering (S/W approach)
- Recency based fingerprint management (S/W approach)
Hardware acceleration technique
• Xilinx Virtex6 XC6VLX240T FPGA
• Verilog HDL 2001 for RTL coding

Components
• Main Controller
  • Govern the logic on the whole
• DIO Controller
  • Data I/O Control unit
  • interfacing the logic with CPU.
• 4KB Dual Port BRAM
  • Storing 4KB data temporary.
• SHA-1 Calculator
  • Generating fingerprints using the standard SHA-1 algorithm.
• Hash Comparator
  • Examines two fingerprints and returns whether they are same or not.
Measurement of fingerprint generation overhead

- Software version of the standard SHA-1 algorithm
  - 150MHz Xilinx Microblaze, 175MHz ARM7, 400MHz ARM9
  - 6212, 5772, 813 us per each SHA-1 calculation, respectively

- SHA-1 hardware logic
  - 80 us on average
  - The deduplication model indicates that, when the programming time is 1300, we can expect the improvement of write latency with the minimum duplication rate of 5%.
Sampling-based Filtering

- Software acceleration technique
  - Selectively applies deduplication for write requests according to their duplicate possibilities

- IRG (Inter Referencing Gap)
  - Time difference between successive duplicate writes

- Red Group
  - most of the IRGs of duplicate writes are less than 500

- Black Group
  - more than 60% of IRGs are less than 4,000
Exploit write buffer in SSD

- OpenSSD
  - Write buffer: 32MB
  - Request: 8000 numbers of 4KB pending write requests can be kept at maximum
Sampling-based Filtering

Algorithm: Three steps

- Sampling
  - extract sample data from a randomly selected offset.

- Bucketing
  - classify write requests into buckets using sampled data as a hash index

- Filtering
  - apply deduplication only to the write requests in the buckets that have multiple requests
Recency-based Fingerprint Management

- To reduce FP manager overhead
  - The characteristics of SSD workloads with a viewpoint of the LRU stack
    - Strong temporal locality
  - Maintains recently generated fingerprints only, rather than managing all generated fingerprints
## Experimental Environments

- **Using Three Boards**

### Hardware configuration

<table>
<thead>
<tr>
<th></th>
<th>OpenSSD</th>
<th>EZ-X5</th>
<th>Xilinx Vertex 6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>ARM7</td>
<td>ARM9</td>
<td>Microblaze</td>
</tr>
<tr>
<td><strong>Clock</strong></td>
<td>175MHz</td>
<td>400MHz</td>
<td>150MHz</td>
</tr>
<tr>
<td><strong>SRAM</strong></td>
<td>96KB</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>DRAM</strong></td>
<td>64MB</td>
<td>64 MB</td>
<td>256MB</td>
</tr>
<tr>
<td><strong>Flash</strong></td>
<td>128GB</td>
<td>64 MB</td>
<td>16GB~</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Main experimental platform</td>
<td>For testing the Sampling-based filtering experiments</td>
<td>For testing the SHA-1 hardware logic experiments</td>
</tr>
</tbody>
</table>

---

- OpenSSD
- EZ-X5
- Xilinx Vertex 6
## Experimental Environments

- 9 applications are used for the experiments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Total written data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows install</td>
<td>Install Microsoft windows XP professional edition</td>
<td>1.6GB</td>
</tr>
<tr>
<td>Linux install</td>
<td>Install Ubuntu 10.10</td>
<td>2.9GB</td>
</tr>
<tr>
<td>Kernel compile</td>
<td>Compile the Linux kernel version 2.6.32</td>
<td>805MB</td>
</tr>
<tr>
<td>Xen compile</td>
<td>Compile the Xen version 4.1.1</td>
<td>634MB</td>
</tr>
<tr>
<td>Office work</td>
<td>Perform Microsoft Office applications (word and power point)</td>
<td>132MB</td>
</tr>
<tr>
<td>Outlook sync</td>
<td>Synchronize Gmail accounts</td>
<td>3.9GB</td>
</tr>
<tr>
<td>HTTrack</td>
<td>Download the contents of Dankook university web site</td>
<td>121MB</td>
</tr>
<tr>
<td>SVN</td>
<td>Using revisions of VirtualBox sources</td>
<td>2.8GB</td>
</tr>
<tr>
<td>Wayback machine</td>
<td>Download archived pages, composed of the first page of the Yahoo! Web site</td>
<td>148MB</td>
</tr>
</tbody>
</table>
Experimental Results

- Duplication Rate

- ranging from 5% to 52% with an average of 17%
  - Among the nine workloads, we can achieve the same duplication rate for each run from the windows install, Linux install, kernel compile and Xen compile workloads.
  - The duplication rate of the office work, HTTrack and outlook workloads depends on the user behavior and contents of a site/mail server.
  - The wayback machine shows the best duplication rate since it writes not only the modified data but also the unchanged data altogether for archiving.
  - The SVN saves modified data only in each revision
Experimental Results

Write latency

- Experimental condition:
  - Use the SHA-1 hardware logic and recency-based FP management
  - Garbage collection is not invoked. (set utilization as 0% before experiments)

- Observation
  - By applying deduplication, we can improve the write latency by up to 48% with an average of 15%.
  - Application with higher duplication rate yields larger benefit
  - The improvements are owing to the elimination of duplicate writes
Experimental Results

- **Write latency**
  - **Experimental condition:**
    - Use the SHA-1 hardware logic and recency-based FP management
    - **Garbage collection is invoked** during application execution (set utilization as 90% before experiments)
  - **Observation**
    - We can improve the write latency by up to 51% with an average of 27%.
    - When the garbage collection is involved, we can get larger performance gain since deduplication can not only eliminate duplicate writes but also reduce the garbage collection overhead.

![Experimental Results Chart](image-url)
Experimental Results

- Duplication rate and write latency
  - Experimental condition:
    - Use the **Sampling-based filtering** and recency-based FP management
    - Garbage collection is invoked during application execution (set utilization as 80% before experiments)
  - Observation
    - Achieving comparable duplication rate to the full FP generation
    - Obtain performance gain without SHA-1 hardware logic
Experimental Results

- **Reliability: WAF and Lifespan**
  - **WAF: Write Amplification Factor**
    - Ratio of data actually written into Flash to data requested by the host
    - Deduplication reduce WAF by reducing both write traffic and GC overhead
  - **Lifespan**
    - Reduction of WAF can lead to enhance the lifetime of SSD
    - From Imation paper (www.csee.umbc.edu/~squire/images/ssd1.pdf)

\[
\text{Lifetime (years)} = \frac{(\text{SSD Capacity (GB)})(P/E)(\text{Percent Utilization})}{(\text{Usage/Day})(\text{Capacity Rate})(365 \text{ days/year})}
\]

<table>
<thead>
<tr>
<th>Utilization</th>
<th>WAF</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>75%</td>
<td>2.9</td>
<td>3.5</td>
</tr>
<tr>
<td>85%</td>
<td>5.5</td>
<td>6.0</td>
</tr>
<tr>
<td>95%</td>
<td>15.8</td>
<td>3.4</td>
</tr>
</tbody>
</table>

- **Expected Lifespan**
  - No deduplication
  - Deduplication

- **Write Amplification Factor**
  - No deduplication
  - Deduplication

![Bar chart showing WAF and Expected Lifespan for different Utilization levels](chart.png)
Conclusions

- Deduplication architecture for SSD
  - FP generator, FP manager, MAP manager

- Deduplication model
  - Minimum duplication rate for performance gain under various deduplication overhead and programming time

- Three acceleration Techniques
  - SHA-1 hardware logic
  - Sampling-based filtering
  - Recency-based fingerprint management

- Experimental results
  - Deduplication is an effective solution for improving the write latency and lifespan of SSDs.
Discussion

- Question or Suggestion