Buffer Flush and Address Mapping Scheme for Flash Memory Solid State Disk

Dongkun Shin
Sungkyunkwan University, Korea
dongkun@skku.edu
Introduction

- Dramatic price reduction of flash memory
- SSD is emerging as a killer application for NAND flash (desktop PC, enterprise server, camcorder)
- **Pros**
  - Low power consumption, high reliability and high random access performance
- **Cons**
  - Expensive cost
- To reduce the cost of SSD,
  - MLC (multi-level cell) flash SSD is a popular recent solution
  - MLC has a slower performance and a shorter life span, making the performance of SSD a critical issue.
Hurdles towards High-Performance

• Slow write performance compared to read performance.
  • Use internal volatile write buffer (SDRAM)
    - long write latency is inevitable when the buffer should be flushed due to its limited capacity.
• Inferior sequential performance compared to HDD
  • Use parallel architecture (multi-channel and multi-way architecture)
    - Program multiple pages on different chips at a time
• Too large mapping information
  • Use coarse-grained mapping such as superblock
    - Large block merge overhead
MLAM

• Two critical issues on designing the NAND flash SSD
  • how to select victim pages for the write buffer flush
  • how to map logical address into physical address considering the parallel architecture of SSD

• Multi-level address mapping technique (MLAM)
  • victim page selection for the write buffer considering the block merge overhead
  • dynamically determines the mapping granularity based on the write pattern
    – Provide fast performance with small mapping table
SSD Architectures

- Park [NVSM’06] : multi-channel and multi-way controller
- Kang [JSA’07] : striping, interleaving and pipelining
- Chang [ASP-DAC’08] : hybrid SSD architecture
- Agrawal [USENIX’08] : trace-driven simulator
  - page-level mapping (async mode)
  - superpage-level mapping (sync mode)
- Shin [ICS’09] : page stripping methods

- No intensive research on the address mapping for flash memory SSD.
Multi-Level Address Mapping

- Wu [ICCAD’05]: two-level address mapping scheme that dynamically switches between page-level and block-level mappings
- Chang [TOS’05]: tree-based management scheme that adopts multiple granularities
- u-FTL [EMSOFT’08]: multi-level mapping managed by u-tree

- No consideration of the parallel handling for interleaved flash chips in SSD
Flash-Aware Buffer Schemes

- CFLRU: delays the flush of dirty pages in buffer cache
- FAB: block-level buffer replacement
- BPLRU: block-level LRU policy and block padding
- REF: considers the recent history on log buffer

- No buffer management scheme considering the parallel architecture of SSD
SSD Internals

- **SDRAM Buffer**: temporally stores data from the host
- **Multi-Channels**: can be accessed simultaneously
- **Multi-Ways**: can be accessed in interleaved manner
- **Superchip**: A group of chips which can be accessed simultaneously.

![Diagram of SSD Internals]

D. Shin@SKKU

NVRAMOS 2009 Fall
Superpage and Superblock

- **Superpage (page group)**
  - A group of pages which can be accessed in parallel
  - All pages have the same offset within a chip

- **Superblock (block group)**
  - Extension of superpage to a group of blocks.
Address Mapping

- Goal: minimize block merge overhead with small mapping table
  - Page mapping: chip selection issue, async or sync, too large map table
  - Superpage mapping (hybrid mapping): fragmentation, large map table
  - Superblock mapping: fragmentation, large SB merge overhead
  - Multi-level mapping
## Mapping Table

### 128GB SSD

<table>
<thead>
<tr>
<th>Mapping Level</th>
<th>Entry Size</th>
<th># of Entry</th>
<th>Total Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page-level</td>
<td>4 bytes</td>
<td>128GB/4KB = 32M</td>
<td>128 MB</td>
</tr>
<tr>
<td>Superpage-level</td>
<td>3 bytes</td>
<td>128GB/32KB = 4096K</td>
<td>12 MB</td>
</tr>
<tr>
<td>Superblock-level</td>
<td>2 bytes</td>
<td>128GB/4MB = 32K</td>
<td>64 KB</td>
</tr>
<tr>
<td>Hybrid-level</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>3 bytes</td>
<td>13GB/32KB= 400K</td>
<td>1.2MB</td>
</tr>
<tr>
<td>Data</td>
<td>2 bytes</td>
<td>115GB/4MB = 29K</td>
<td></td>
</tr>
</tbody>
</table>

Page: 4KB  
Superpage: 32KB  
Superblock: 4MB  
Hybrid: log buffer is 10% of total storage
Superpage-Level Mapping

- Small mapping table compared to page-level mapping, but still too large in large-scaled SSD
- Fragmentation (there are unused pages)
- Requires copyback for unmodified pages

\[(\text{LPN} \mod N_{\text{chip}}) = \text{ChipID} \] (in-place for all pages)

D. Shin@SKKU  NVRAMOS 2009 Fall
Superblock-Level Mapping

- Small mapping table
- Large fragmentation
- Superblock merge overhead for small-sized requests
Log Buffer Useful in SSD?

- Superpage-level or hybrid-level mapping will be more efficient than superblock-level mapping if a workload has **high temporal locality and low spatial locality** (random pattern).

- However, write requests on flash chips come through several buffers, which perform merging and sorting for small-sized write requests.

- Therefore, they have little temporal locality but high spatial locality (due to buffer's merging operation).

- How about multiple mapping granularity?

- But arbitrary mapping granularities require high complexity (eg. u-FTL)
Sub-Superblock

1/2^n superblocks or 2^{7-n} superpages (0 ≤ n ≤ m)
Multi-Level Mapping

Find the largest mapping unit which invokes a merge overhead less than the predefined portion.

D. Shin@SKKU

NVRAMOS 2009 Fall
Virtual Superblock Composition

- Sub-superblock writing invokes the fragmentations within PSB
- Write by the unit of PSB
- Compose one virtual superblock with several sub-superblocks and write the VSB at a PSB
- We need several victim logical superblocks to compose a VSB
Victim LSB Selection

- SIZE policy
  - Choose the biggest LSB which means that most data are to be updated.
  - Small-sized LSB could remain without being flushed.
- LRU policy
  - Choose the LSB which has not been accessed for the longest time.
  - Old and small-sized LSB may deteriorate performance.
- LRU+Size policy
  - Consider both two factors
    \[ Pr(B_i) = \alpha \cdot \frac{t(B_i)}{T} + (1 - \alpha) \cdot \frac{n_{\text{page}}(B_i)}{N} \]
Virtual Superblock Composition

- Each victim LSB is partitioned into sub-SBs if it has more than $k_{\text{empty}}$ empty blocks.
- Group the victim sub-SBs based on the superchip index.
- Compose a VSB for each superchip such that it has the largest number of updated pages.
- Select the largest-sized VSB among the VSBs for several superchips.

D. Shin@SKKU

NVRAMOS 2009 Fall
Multi-Level Address Mapping

Flash memory (before eviction of V)

Flash memory (after eviction of V)

LSB  | psb | tab
--- | --- | ---
0    | 10  | 00
1    | 20  | 00
2    | 154 | 00
3    | 162 | 00
4    |     |   

L₀ mapping table (initial state)

PSB#  | sub-superblock index
--- | ---
10   | 0 1 2 3
20   | 0 1 2 3
154  | 0 1 2 3
162  | 0 1 2 3

B₄ B₀ B₁

VSB

sub-superblock index

invalidated

write

LSB  | psb₀ | psb₁ | tab₀ | tab₁
--- | --- | --- | --- | ---
0    | 0   | 0   | 2   | 1
1    | 1   | 2   | 1   | 1
2    | 154 | 0   | 0   | 0
3    | 162 | 0   | 0   | 0
4    | 1   | 2   |     |   

index | psb | loc
--- | --- | ---
0    | 10  | 1
1    | 20  | 0
2    | 56  | 1

L₁ mapping table

index | psb₀ | loc₀ | psb₁ | loc₁
--- | --- | --- | --- | ---
0    | 10  | 0   | 56  | 1
1    | 56  | 0   |     |   

L₂ mapping table

D. Shin@SKKU  NVRAMOS 2009 Fall
Experiments

- Our SSD simulator
- 4-channel and 2-way
- 16~128 MB SDRAM
- 32 1GB MLC flash chips
- 5 real disk I/O traces and 1 benchmark trace

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page size</td>
<td>4KB</td>
<td>Page read</td>
<td>60 (\mu s)</td>
</tr>
<tr>
<td>Block size</td>
<td>512KB</td>
<td>Page write</td>
<td>800 (\mu s)</td>
</tr>
<tr>
<td>(128 pages)</td>
<td></td>
<td>Block erase</td>
<td>1.5 ms</td>
</tr>
<tr>
<td>Superpage size</td>
<td>32KB</td>
<td>Page copyback</td>
<td>860 (\mu s)</td>
</tr>
<tr>
<td>Superblock size</td>
<td>4096KB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NVRAMOS 2009 Fall
Experiments

- Mapping level comparison with varying $k_{\text{empty}}$

(a) Desktop

(b) pcFAT32

(c) pcNTFS

(d) JPEG File Copy

(e) Internet Explorer

(f) lozone

same to superblock mapping
Experiments

- Execution time comparison with varying $k_{\text{empty}}$
Experiments

- Execution time comparison with varying $k_{\text{empty}}$
Experiments

- Execution time comparison while varying the buffer size

![Graphs showing execution time comparison for different buffer sizes in various environments: Desktop, pcFAT32, pcNTFS, JPEG File Copy, Internet Explorer, and Tozone. Each graph compares times for Erase, RNW, Copyback, Write, and Read operations across different buffer sizes (16MB, 32MB, 64MB, 128MB).]
Experiments

• Comparison between victim selection policies

(a) Desktop

(b) pcFAT32

(c) pcNTFS

(d) JPEG File Copy

(e) Internet Explorer

(f) Izone

Time (s)

FlushAll LRU Size LRUSize MLAM

Time (s)

FlushAll LRU Size LRUSize MLAM

Time (s)

FlushAll LRU Size LRUSize MLAM

Time (s)

FlushAll LRU Size LRUSize MLAM

Time (s)

FlushAll LRU Size LRUSize MLAM

D. Shin@SKKU

NVRAMOS 2009 Fall

28
Conclusions

• The parallel architecture (multi-channel and multi-way) is essential to the high performance NAND flash SSD.

• The coarse-grained mapping can show poor performance when there are many random and scattered write requests.

• Can reduce the superblock merge overhead significantly by allowing multi-level mappings.