Flash Translation Layers III

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Contents

- Superblock FTL
- DFTL
- μ-FTL
Superblock FTL

Reference: Superblock FTL: A Superblock-based Flash Translation Layer with a Hybrid Address Translation Scheme
Superblock FTL

- A superblock shares log blocks
- Up to M log blocks per superblock
- Block mapping at the superblock level, Page mapping within a superblock
- Hot/cold pages separation
- Map cache
- The amount of mapping information increased
Example

- $W = \langle \{1,2\}, \{8\}, \{1,2\}, \{12\}, \{13\}, \{9\} \rangle$
  - Write ($\{1,2\}$, AB)
  - Write ($\{8\}$, C)
  - Write ($\{1,2\}$, DE)
  - Write ($\{12\}$, F)
  - Write ($\{13\}$, G)
  - Write ($\{9\}$, H)
Mapping Information

- PGD (Page Global Directory)
- PMD (Page Middle Directory)
- PTE (Page Table Entry)
## Comparison

<table>
<thead>
<tr>
<th>Data blocks</th>
<th>ANAND Terminology</th>
<th>Log block scheme</th>
<th>FAST Data blocks</th>
<th>Superblock FTL D-blocks</th>
<th>Page mapping Data blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management scheme</td>
<td>In-order</td>
<td>In-order</td>
<td>In-order</td>
<td>Out-of-order</td>
<td>Out-of-order</td>
</tr>
<tr>
<td>The degree of sharing</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Min(P, S)</td>
<td>P</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Update blocks</th>
<th>Terminology Replacement blocks</th>
<th>Log blocks</th>
<th>Sequential / random log blocks</th>
<th>U-blocks</th>
<th>Update blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management scheme</td>
<td>In-order</td>
<td>Out-of-order</td>
<td>Out-of-order</td>
<td>Out-of-order</td>
<td>Out-of-order</td>
</tr>
<tr>
<td>The degree of sharing</td>
<td>1</td>
<td>1</td>
<td>1 (sequential) or P (random)</td>
<td>Min(P, S)</td>
<td>P</td>
</tr>
</tbody>
</table>
Reference: DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings
DFTL

- Page mapping
- Keeps full mapping on flash to reduce SRAM use
- Demand-based selective caching of page-level address mappings in SRAM
- Exploits temporal locality in workloads
- Data pages vs. Translation pages
DFTL Architecture

Stores active address mappings

Cached Mapping Table

Consult location of translation pages on flash

Global Translation Directory

SRAM

Tracks translation pages on flash

Store logical to physical address translations

Fetch mapping entry

Evict mapping entry for Synchronization

Store real data from I/O requests

Translation Blocks

FLASH

Data OOB

Data Blocks
Garbage Collection

- Current data block: updated data pages
- Current translation block: updated translation pages
- GC invoked if the number of free blocks < GC_threshold
- Selecting a victim block: simple cost-benefit policy
μ-FTL

Reference: μ-FTL: A Memory-efficient Flash Translation Layer Supporting Multiple Mapping Granularities
μ-FTL

- Minimally-updated FTL
- Supports multiple mapping granularities
  - Based on extents
  - Reduce the amount of mapping information
- Requires more sophisticated index structure
  - μ-Tree is used to store the mapping info.
- Tunable memory footprint
  - Frequently accessed mapping info. cached
Example

- $W = \langle \{1\}, \{2\}, \{8\}, \{1\}, \{2\}, \{12,13\}, \{9\} \rangle$
  - write ($\{1\}$, A)
  - write ($\{2\}$, B)
  - write ($\{8\}$, C)
  - write ($\{1\}$, D)
  - write ($\{2\}$, E)
  - write ($\{12,13\}$, FG)
  - write ($\{9\}$, H)
μ-FTL Architecture

μ-Tree Cache

Bitmap update flush

Bitmap Cache

Write

Extent update

Data

Cache full

Cache miss

μ-Tree

Partition 1

Partition 2

Partition 3

Partition 4

Free block list
Conclusion

- Garbage collection issues
- Mapping information management issues
- Platform-dependent issues
CAFTL
(Content-Aware FTL)

Reference: CAFTL - A Content-Aware FTL Enhancing the Lifespan of Flash Memory based Solid State Drives (2011)
CAFTL

- Deduplication
  - Reduce write traffic to flash memory by removing unnecessary duplicate writes

![Graph](image)

**Figure 1:** The percentage of redundant data in disks.
Identify duplicate blocks

- **Fingerprint**
  - Byte-by-byte comparison is slow
  - Identify duplicate blocks by comparing SHA-1 hash values

- **Fingerprint Store**
  - Store and search in the most likely-to-be-duplicated fingerprints
Architecture

Figure 3: An illustration of CAFTL architecture.
Two-level indirect mapping

Mapping Table

Flash Memory

LPN 0
LPN 1
LPN 2
LPN 3
LPN 4
LPN 5
LPN 6
LPN 7
LPN 8
LPN 9
LPN 10
LPN 11
LPN 12
LPN 13
LPN 14
LPN 15

A
F
T
I

L
L
C
T

T
F
E
H

W
R
B
Z

Conventional FTL

Primary Map

Secondary Map

Flash Memory

VPN 0
VPN 1
VPN 2
VPN 3
VPN 4
VPN 5

VPN 0
VPN 1
VPN 2
VPN 3
VPN 4
VPN 5

F
A
E
T

H
Z
L
R

C
B
I
W

CAFTL
Example (1)

- \( W = \langle \{1\}, \{2\}, \{0\}, \{11\}, \{4\}, \{7\} \rangle \)
  - write \( (\{1\}, F) \)
  - write \( (\{2\}, F) \)
  - write \( (\{0\}, A) \)
  - write \( (\{11\}, E) \)
Example (2)

- \( W = \langle \{1\}, \{2\}, \{0\}, \{11\}, \{4\}, \{7\} \rangle \)
  - write \( \{1\}, \text{F} \)
  - write \( \{2\}, \text{F} \)
  - write \( \{0\}, \text{A} \)
  - write \( \{11\}, \text{E} \)
  - write \( \{4\}, \text{T} \)
Example (3)

- \( W = \langle \{1\}, \{2\}, \{0\}, \{11\}, \{4\}, \{7\} \rangle \)
  - write (\( \{1\}, F \))
  - write (\( \{2\}, F \))
  - write (\( \{0\}, A \))
  - write (\( \{11\}, E \))
  - write (\( \{4\}, T \))
  - write (\( \{7\}, T \))
Performance Evaluation

Figure 8: Perc. of removed duplicate writes.

Figure 9: Perc. of extended flash space.