Flash Translation Layers II

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

- **Mapping in block-level**
  - Logical block number $\rightarrow$ physical block number
  - Block mapping table (BMT) required
  - Page offset remains the same

- **Translation**
  - Step 1: logical sector number $\rightarrow$ logical page number (LPN)
  - Step 2: LPN $\rightarrow$ logical block number (LBN) + page offset
  - LBN = LPN / N, where N: # of pages in a block
  - Page offset = LPN % N
  - Step 3: LBN $\rightarrow$ physical block number (PBN) via BMT
    - Use the same page offset
Page vs. Block Mapping

Page mapping

Block mapping
Example: Block Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4

- Current state
  - Written to page 0, 1, 2, 8, 4, 5

- Reading page 5
Example: Block Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4

- Current state
  - Written to page 0, 1, 2, 8, 4, 5

- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5 (update)
Example: Block Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4

- Current state
  - Written to page 0, 1, 2, 8, 4, 5

- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5 (update)
Example: Block Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4

- Current state
  - Written to page 0, 1, 2, 8, 4, 5

- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5 (update)
Example: Block Mapping

- **Flash configuration**
  - Page size: 4KB
  - # of pages / block = 4

- **Current state**
  - Written to page 0, 1, 2, 8, 4, 5

- **New requests (in order)**
  - Write to page 9
  - Write to page 3
  - Write to page 5 (update)
Example: Block Mapping

- **Current state**
  - Written to page 0, 1, 2, 8, 4, 5
  - Written to page 9, 3, 5

- **New requests (in order)**
  - Write to page <0, 1, 2, 3>
Example: Block Mapping

- **Current state**
  - Written to page 0, 1, 2, 8, 4, 5
  - Written to page 9, 3, 5

- **New requests (in order)**
  - Write to pages <0, 1, 2, 3>
Block Mapping

▪ Pros
  • Small RAM usage
    – One mapping entry per block
  • Good performance for sequential writes

▪ Cons
  • Inefficient handling of small random writes
    – Even a single page update requires a block copy & erase
Hybrid Mapping

- Exploits both mapping schemes
  - Page mapping
    - Update block or log block
  - Block mapping
    - Data blocks

<table>
<thead>
<tr>
<th>Pros</th>
<th>Page Mapping</th>
<th>Block Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Efficient handling of small writes</td>
<td>• Small management overhead for translation information</td>
</tr>
<tr>
<td>Cons</td>
<td>• Large management overhead for translation information</td>
<td>• Inefficient handling of small writes</td>
</tr>
</tbody>
</table>
BAST (or Log Block Scheme)

J. Kim et al., “A Space-efficient Flash Translation Layer for Compact Flash Systems,”
BAST

- One of early versions of hybrid mapping
- Key ideas
  - Provide log blocks for storing updates
    - Data blocks are managed in block mapping
    - Log blocks are managed in page mapping
  - 1-to-1 mapping between log block and data block
    - One log block is dedicated to a single data block
  - Mix fine- and coarse-grained management schemes
- Homogeneous log blocks
  - Each log block is the same in type and role (see FAST for different example)
Data vs. Log Blocks

- **Data Block**
  - Block level mapping

- **Log Block**
  - Page level mapping
  - Temporary block associated with the data block

\[ W = \langle \{0\}, \{1,2\}, \{10\}, \{9\} \rangle \]
Block Merge

- Block merge happens when
  - All the pages of a log block are overwritten
  - No available log block to allocate to a data block

- Types of block merge
  - Switch / Partial / Full
  - Cost: Switch < Partial < Full
  - System maintains at least one free block for full merge operation
Switch Merge (1)

- \( W = \langle \{0\}, \{1,2\}, \{10\}, \{9\}, \{3\} \rangle \)
  - Write (\( \{0\}, A \))
  - Write (\( \{1,2\}, BC \))
  - Write (\( \{10\}, D \))
  - Write (\( \{9\}, E \))
Switch Merge (2)

- \( W = \langle \{0\}, \{1,2\}, \{10\}, \{9\}, \{3\} \rangle \)
  - Write (\( \{0\}, A \))
  - Write (\( \{1,2\}, BC \))
  - Write (\( \{10\}, D \))
  - Write (\( \{9\}, E \))
  - Write (\( \{3\}, F \))
Switch Merge (3)

- \( W = \langle \{0\}, \{1,2\}, \{10\}, \{9\}, \{3\} \rangle \)
  - Write (\{0\}, A)
  - Write (\{1,2\}, BC)
  - Write (\{10\}, D)
  - Write (\{9\}, E)
  - Write (\{3\}, F)
Partial Merge (1)

- $W = \langle \{0\}, \{1,2\}, \{10\}, \{9\} \rangle$
  - Write ($\{0\}, A$)
  - Write ($\{1,2\}, BC$)
  - Write ($\{10\}, D$)
  - Write ($\{9\}, E$)
Partial Merge (2)

- $W = \langle \{0\}, \{1,2\}, \{10\}, \{9\} \rangle$
  - Write (\{0\}, A)
  - Write (\{1,2\}, BC)
  - Write (\{10\}, D)
  - Write (\{9\}, E)
Partial Merge (3)

- \( W = \langle \{0\}, \{1,2\}, \{10\}, \{9\} \rangle \)
  - Write (\( \{0\}, A \))
  - Write (\( \{1,2\}, BC \))
  - Write (\( \{10\}, D \))
  - Write (\( \{9\}, E \))
Full Merge (1)

- \( W = \langle\{0\}, \{1,2\}, \{10\}, \{9\}, \{12\}\rangle \)
  - Write (\(\{0\}, A\))
  - Write (\(\{1,2\}, BC\))
  - Write (\(\{10\}, D\))
  - Write (\(\{9\}, E\))
Full Merge (2)

- \( W = \langle \{0\}, \{1,2\}, \{10\}, \{9\}, \{12\} \rangle \)
  - Write (\{0\}, A)
  - Write (\{1,2\}, BC)
  - Write (\{10\}, D)
  - Write (\{9\}, E)
Full Merge (3)

- \( W = \langle \{0\}, \{1,2\}, \{10\}, \{9\}, \{12\} \rangle \)
  - Write \( \{0\}, A \)
  - Write \( \{1,2\}, BC \)
  - Write \( \{10\}, D \)
  - Write \( \{9\}, E \)
Full Merge (4)

- $W = \langle \{0\}, \{1,2\}, \{10\}, \{9\}, \{12\} \rangle$
  - Write ($\{0\}, A$)
  - Write ($\{1,2\}, BC$)
  - Write ($\{10\}, D$)
  - Write ($\{9\}, E$)
Full Merge (5)

- $W = \langle \{0\}, \{1,2\}, \{10\}, \{9\}, \{12\} \rangle$
  - Write ($\{0\}$, A)
  - Write ($\{1,2\}$, BC)
  - Write ($\{10\}$, D)
  - Write ($\{9\}$, E)
  - Write ($\{12\}$, F)
Types of Block Merges

- **Full Merge**
  - (two block erases, Max. Np page copies)

- **Partial Merge**
  - (one block erase, Max. Np - 1 page copies)

- **Switch Merge**
  - (one block erase, No page copy)

```plaintext
<table>
<thead>
<tr>
<th>Data block</th>
<th>Free Block</th>
<th>Log block</th>
<th>Data block</th>
<th>Log block</th>
<th>Data block</th>
<th>Log block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid</td>
<td>Free</td>
<td>Invalid</td>
<td>Invalid</td>
<td>Valid</td>
<td>Invalid</td>
<td>Valid</td>
</tr>
<tr>
<td>Invalid</td>
<td>Free</td>
<td>Valid</td>
<td>Invalid</td>
<td>Valid</td>
<td>Invalid</td>
<td>Valid</td>
</tr>
<tr>
<td>Valid</td>
<td>Free</td>
<td>Invalid</td>
<td>Valid</td>
<td>Free</td>
<td>Invalid</td>
<td>Valid</td>
</tr>
<tr>
<td>Valid</td>
<td>Free</td>
<td>Valid</td>
<td>Valid</td>
<td>Free</td>
<td>Valid</td>
<td>Valid</td>
</tr>
</tbody>
</table>

```
Block State Transition

- **Free Block**
  - Write
  - Erase

- **Log Block**
  - Write
  - Switch merge
  - Partial merge

- **Data Block**
  - Full merge
  - Partial merge

- **Erasable Block**
  - Full merge
  - Switch merge
  - Partial merge

- States:
  - Valid
  - Free
  - Invalid
  - Valid
BAST Performance

- Simulation results

<table>
<thead>
<tr>
<th>R</th>
<th>Replacement block scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Log block scheme (BAST)</td>
</tr>
<tr>
<td>L-M</td>
<td>Log block scheme without map blocks</td>
</tr>
<tr>
<td>P/G</td>
<td>Page-level scheme with a greedy policy</td>
</tr>
<tr>
<td>P/C</td>
<td>Page-level scheme with a cost-benefit policy</td>
</tr>
</tbody>
</table>

Canon PowerShot G1

Kodak DC290
BAST

- **Pros**
  - Better performance than pure block-mapping scheme
  - Reduced resource requirements for page mapping management

- **Cons**
  - Low utilization of log blocks
    - Log block utilization = (# of written pages) / (# of pages in log block)
    - If the write pattern is random, log block is frequently merged despite its low utilization
Problems of BAST

- Low space utilization of log blocks when they are replaced from the log buffer
  - Space utilization: the percentage of the written pages in a log block when it is being replaced

- Another view of the log block scheme
  - Block-Associative Sector Translation approach (BAST)
  - The scheme directs all the overwrites for one logical block only to its dedicated log block
Problems of BAST

- **Log block thrashing**
  - High miss ratio in cache
  - When replaced, some pages remain empty
  - e.g. Write sequence: 0, 4, 8, 12, 0, 4, 8, 12, … \((Np = 4)\)

- **Intensive overwrites**
  - The log block scheme might result in increased write operations
  - e.g. Write sequence: 0, 2, 1, 3, 1, 0, 2, 3, … \((Np = 4)\)
Problems of BAST

- Simulation results on space utilization
  - For a fixed number of log blocks, the space utilization gets worse as the write pattern becomes more random (number of log blocks < number of hot data blocks)
  - As more log blocks are available, each log block can stay longer
FAST

- Fully-Associative Sector Translation
  - Overcome the problems of BAST
    - Alleviate log block thrashing
    - Avoid frequent merge operations
  - N-to-M mapping between data blocks and log blocks
    - Fully associative approach in mapping logical pages onto log blocks

![Diagram: Comparison between BAST and FAST](image-url)
FAST Architecture

- Two types of log blocks
  - **SW**: Sequential Write log block (just one)
    - To increase the chance of switch merge for sequential writes
  - **RW**: Random Write log block

- Mapping table
  - Block-level mapping table (for data blocks)
  - Page-level mapping table (for SW)
  - Page-level mapping table (for RW)
Handling Writes

- Written to SW log block if
  - LPN mod \( Np \) == 0 (LPN is at the beginning of a block) or
  - LPN immediately follows the pages written in SW log block

- Cases 1 & 2.1:
  - Inserted into SW log block

- Cases 2.2 & 2.3:
  - Merge SW log block with its data block

- Case 3:
  - Inserted into RW log block
  - Select victim in round-robin fashion if needed
### Merge: SW Log Block

- **SW log block results in switch or partial merge**

#### Data block

<table>
<thead>
<tr>
<th>PBN=9</th>
<th>...</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Write(8,···)**
- **Write(9,···)**
- **Write(10,···)**
- **Write(11,···)**
- **Switch Merge**

<table>
<thead>
<tr>
<th>PBN=15</th>
</tr>
</thead>
</table>

- **Partial Merge**

<table>
<thead>
<tr>
<th>PBN=9</th>
<th>...</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| PBN=15 |

- **Copy**

---

ICE3028: Embedded Systems Design | Spring 2017 | Jin-Soo Kim (jinsookim@skku.edu)
Without SW Log Block (1)

- \( W = \langle \{7\}, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\} \rangle \)
  - Write (\( \{7\}, A \))
  - Write (\( \{0,1,2\}, \text{BCD} \))
  - Write (\( \{5\}, E \))
  - Write (\( \{3\}, F \))
  - Write (\( \{11\}, G \))
  - Write (\( \{15\}, H \))
Without SW Log Block (2)

- $W = \langle \{7\}, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\} \rangle$
  - Write ($\{7\}$, A)
  - Write ($\{0,1,2\}$, BCD)
  - Write ($\{5\}$, E)
  - Write ($\{3\}$, F)
  - Write ($\{11\}$, G)
  - Write ($\{15\}$, H)
Without SW Log Block (3)

- \( W = \langle \{7\}, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\} \rangle \)
  - Write (\( \{7\}, A \))
  - Write (\( \{0,1,2\}, BCD \))
  - Write (\( \{5\}, E \))
  - Write (\( \{3\}, F \))
  - Write (\( \{11\}, G \))
  - Write (\( \{15\}, H \))
Without SW Log Block (4)

- \( W = \langle \{7\}, \{0, 1, 2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\} \rangle \)
  - Write (\( \{7\}, A \) )
  - Write (\( \{0, 1, 2\}, \text{BCD} \) )
  - Write (\( \{5\}, E \) )
  - Write (\( \{3\}, F \) )
  - Write (\( \{11\}, G \) )
  - Write (\( \{15\}, H \) )
Without SW Log Block (5)

- $W = \langle \{7\}, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\} \rangle$
  - Write (\{7\}, A)
  - Write (\{0,1,2\}, BCD)
  - Write (\{5\}, E)
  - Write (\{3\}, F)
  - Write (\{11\}, G)
  - Write (\{15\}, H)
  - Write (\{0\}, I)
With SW Log Block (1)

- \( W = \langle \{7\}, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\} \rangle \)
  - Write (\( \{7\}, A \))
  - Write (\( \{0,1,2\}, BCD \))
  - Write (\( \{5\}, E \))
  - Write (\( \{3\}, F \))
With SW Log Block (2)

- $W = \langle \{7\}, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\}\rangle$
  - Write ($\{7\}$, $A$)
  - Write ($\{0,1,2\}$, BCD)
  - Write ($\{5\}$, E)
  - Write ($\{3\}$, F)
  - Write ($\{11\}$, G)
  - Write ($\{15\}$, H)
With SW Log Block (3)

- $W = \langle \{7\}, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\}\rangle$
  - Write ($\{7\}, A$)
  - Write ($\{0,1,2\}, BCD$)
  - Write ($\{5\}, E$)
  - Write ($\{3\}, F$)
  - Write ($\{11\}, G$)
  - Write ($\{15\}, H$)
  - Write ($\{0\}, I$)
With SW Log Block (4)

- \( W = \langle 7, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\} \rangle \)
  - Write \( \{7\}, A \)
  - Write \( \{0,1,2\}, BCD \)
  - Write \( \{5\}, E \)
  - Write \( \{3\}, F \)
  - Write \( \{11\}, G \)
  - Write \( \{15\}, H \)
  - Write \( \{0\}, I \)
**Merge: RW Log Block**

- The victim selection is done in a round-robin fashion
  - Pages in a victim originate from several different logical blocks
  - Usually, more than one full merge required

```
0 1 2 3  4 5 6 7  8 9 10 11
1 8 3 9  3 8 10 2
```
The victim selection is done in a round-robin fashion

- Pages in a victim originate from several different logical blocks
- Usually, more than one full merge required

1. Select victim
The victim selection is done in a round-robin fashion

- Pages in a victim originate from several different logical blocks
- Usually, more than one full merge required

1. Select victim
2. Scan for most up-to-date data for LBN 0
The victim selection is done in a round-robin fashion
- Pages in a victim originate from several different logical blocks
- Usually, more than one full merge required
Merge: RW Log Block

- The victim selection is done in a round-robin fashion
  - Pages in a victim originate from several different logical blocks
  - Usually, more than one full merge required

1. Select victim
2. Scan for most up-to-date data for LBN 0
3. Full merge for LBN 0
4. Scan for most up-to-date data for LBN 2
Merge: RW Log Block

- The victim selection is done in a round-robin fashion
  - Pages in a victim originate from several different logical blocks
  - Usually, more than one full merge required

```
1. Select victim
2. Scan for most up-to-date data for LBN 0
3. Full merge for LBN 0
4. Scan for most up-to-date data for LBN 2
5. Full merge for LBN 2
```
Merge: RW Log Block

- The victim selection is done in a round-robin fashion
  - Pages in a victim originate from several different logical blocks
  - Usually, more than one full merge required

1. Select victim
2. Scan for most up-to-date data for LBN 0
3. Full merge for LBN 0
4. Scan for most up-to-date data for LBN 2
5. Full merge for LBN 2
6. Erase blocks
O-FAST

- Lazy merge
  - Delay merge for a data block if more recent versions exist in the non-victim log blocks

```
0 1 2 3
4 5 6 7
8 9 10 11
1 8 3 10
3 8 10 2
```
O-FAST

- Lazy merge
  - Delay merge for a data block if more recent versions exist in the non-victim log blocks

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

1. Select victim
**O-FAST**

- **Lazy merge**
  - Delay merge for a data block if more recent versions exist in the non-victim log blocks

1. Select victim
2. Scan for most up-to-date data for LBN 0
O-FAST

- Lazy merge
  - Delay merge for a data block if more recent versions exist in the non-victim log blocks

1. Select victim
2. Scan for most up-to-date data for LBN 0
3. Full merge for LBN 0
O-FAST

- Lazy merge
  - Delay merge for a data block if more recent versions exist in the non-victim log blocks

1. Select victim
2. Scan for most up-to-date data for LBN 0
3. Full merge for LBN 0
4. Scan for most up-to-date data for LBN 2
O-FAST

- Lazy merge
  - Delay merge for a data block if more recent versions exist in the non-victim log blocks

1. Select victim
2. Scan for most up-to-date data for LBN 0
3. Full merge for LBN 0
4. Scan for most up-to-date data for LBN 2
5. Skip merging of LBN 2
O-FAST

- Lazy merge
  - Delay merge for a data block if more recent versions exist in the non-victim log blocks

1. Select victim
2. Scan for most up-to-date data for LBN 0
3. Full merge for LBN 0
4. Scan for most up-to-date data for LBN 2
5. Skip merging of LBN 2
6. Erase blocks
FAST Performance

Pattern A, B
Small random and large sequential writes

Pattern C, D
Small random and small large sequential writes

Pattern E
Uniform random writes
FAST

- **Pros**
  - Improved log block utilization
  - Avoid unnecessary merge

- **Cons**
  - Increased merge time
  - Detecting sequential writes is hard
  - Looking up the page-level mapping table for RW log blocks
Superblock FTL

Superblock

- Superblock
  - A set of adjacent logical blocks
  - A superblock shares log blocks
  - Up to M log blocks per superblock
- Hybrid mapping
  - Block mapping at the superblock level
  - Page mapping within a superblock
- Hot/cold data separation within a superblock
  - Gathers hot pages into the same block
- Map cache
Block-level Locality

- **Block-level spatial locality**
  - The pages in the adjacent logical blocks are likely to be updated in the near future
  - When two or more adjacent logical blocks are allocated by file systems to the same file or to the same metadata
  - Make a superblock share data blocks and log blocks
  - Superblock controls the “degree of sharing”

- **Block-level temporal locality**
  - Pages in the same logical block are likely to be updated again in the near future
  - Allocate more than one log block to each superblock
Example: Superblock

- **Configurations**
  - # pages / block = 4
  - Superblock size: 2
  - SBN = LPN / 8

- **Current state**
  - LPNs 0-15 are sequentially written to PPNs 0-15
Example: Superblock

- New requests
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- **New requests**
  - Write \(<1, 2>\)
  - Write 8
  - Write \(<1, 2>\)
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- **New requests**
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8

```
SBN  PBNs
---  ---
  0   4
  1   5
```

```
SB Map Table
0
1
```

```
Log Block Map Table
  0
  1
  2
  3
  4
  5
  6
  7
```

```
Data Blocks
  PBN: 4
  0
  1
  2
  3
  4
  5
  6
  7
```

```
PBN: 5
  1
  2
  3
  4
  5
  6
  7
```

ICE3028: Embedded Systems Design | Spring 2017 | Jin-Soo Kim (jinsookim@skku.edu)
Example: Superblock

- New requests
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- New requests
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- New requests
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- New requests
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- New requests
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- **New requests**
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8

![Superblock diagram with new requests and data blocks]
Example: Superblock

- **New requests**
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- **New requests**
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Example: Superblock

- New requests
  - Write <1, 2>
  - Write 8
  - Write <1, 2>
  - Write 12
  - Write 13
  - Write 9
  - Write 8
Address Translation
Mapping Information

- PGD (Page Global Directory)
- PMD (Page Middle Directory)
- PTE (Page Table Entry)
Superblock Performance

- Superblock size: 4
- Map cache entries: 16 (spare areas)
  - Map cache hit ratio: > 90%
Superblock

Pros

• Flexible within a superblock
• Reduces merge count while merge time kept low

Cons

• The amount of mapping information increased
  – Separate map page required for MLCs
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>BAST</th>
<th>FAST</th>
<th>Superblock</th>
<th>Page mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data blocks</strong></td>
<td>Terminology</td>
<td>Data blocks</td>
<td>Data blocks</td>
<td>D-blocks</td>
</tr>
<tr>
<td>Management scheme</td>
<td>In-order</td>
<td>In-order</td>
<td>Out-of-order</td>
<td>Out-of-order</td>
</tr>
<tr>
<td><strong>Max degree of sharing</strong></td>
<td>1</td>
<td>1</td>
<td>Min(P, S)</td>
<td>P</td>
</tr>
<tr>
<td><strong>Update blocks</strong></td>
<td>Terminology</td>
<td>Log blocks</td>
<td>SW / RW log blocks</td>
<td>U-blocks</td>
</tr>
<tr>
<td><strong>Max degree of sharing</strong></td>
<td>1</td>
<td>1 (sequential) or P (random)</td>
<td>Min(P, S)</td>
<td>P</td>
</tr>
<tr>
<td><strong>Block merge</strong></td>
<td>Frequency</td>
<td>Middle</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Average Cost</td>
<td>Middle</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Average Cost**

Middle

Low

Low