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 → physical block number
  - Block mapping table (BMT) required
  - Page offset remains the same

- Translation
  - Step 1: logical sector number → logical page number (LPN)
  - Step 2: LPN → logical block number (LBN) + page offset
    - LBN = LPN / N, where N: # of pages in a block
    - Page offset = LPN % N
  - Step 3: LBN → 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**

![Block Mapping Diagram]

**Logical block number**

**Page offset within a block**

**Logical page #5**

00000001 01
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></th>
<th>Page Mapping</th>
<th>Block Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td>• Efficient handling of small writes</td>
<td>• Small management overhead for translation information</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>• Large management overhead for translation information</td>
<td>• Inefficient handling of small writes</td>
</tr>
</tbody>
</table>
BAST (or Log Block Scheme)

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

\[ W = \{0\}, \{1,2\}, \{10\}, \{9\} \]
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 $\langle \{0\}, A \rangle$
  - Write $\langle \{1,2\}, BC \rangle$
  - Write $\langle \{10\}, D \rangle$
  - Write $\langle \{9\}, E \rangle$
  - Write $\langle \{3\}, F \rangle$
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 $\langle \{0\}, A \rangle$
  - Write $\langle \{1,2\}, BC \rangle$
  - Write $\langle \{10\}, D \rangle$
  - Write $\langle \{9\}, E \rangle$
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

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

#### Full Merge
(two block erases, Max. $N_p$ page copies)

#### Partial Merge
(one block erase, Max. $N_p - 1$ page copies)

#### Switch Merge
(one block erase, No page copy)
BAST Performance

- Simulation results

| R | Replacement block scheme |
| L | Log block scheme (BAST)   |
| L-M | Log block scheme without map blocks |
| P/G | Page-level scheme with a greedy policy |
| P/C | Page-level scheme with a cost-benefit policy |

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, … \( (\mathcal{N}_p = 4) \)

- Intensive overwrites
  - The log block scheme might result in increased write operations
  - e.g. Write sequence: 0, 2, 1, 3, 1, 0, 2, 3, … \( (\mathcal{N}_p = 4) \)
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

BAST (1:1 mapping)  FAST (N:M mapping)
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 $N_p$ == 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

<table>
<thead>
<tr>
<th>Data block</th>
<th>PBN=9</th>
<th>PBN=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td><strong>SW log block</strong></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>Data block</td>
<td>PBN=9</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td><strong>Switch Merge</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data block</th>
<th>PBN=9</th>
<th>PBN=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td><strong>SW log block</strong></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>Data block</td>
<td>PBN=15</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Partial Merge</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Switch Merge
- Partial Merge
Without 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 (\( \{3\}, F \))
  - 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 (\( \{0,1,2\}, \text{BCD} \))
  - 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 (\(\{0,1,2\}, BCD\))
  - 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 \( \langle 7, A \rangle \)
  - Write \( \langle 0,1,2, \text{BCD} \rangle \)
  - Write \( \langle 5, E \rangle \)
  - Write \( \langle 3, F \rangle \)
  - Write \( \langle 11, G \rangle \)
  - Write \( \langle 15, H \rangle \)
  - Write \( \langle 0, I \rangle \)
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** = <\{7\}, \{0,1,2\}, \{5\}, \{3\}, \{11\}, \{15\}, \{0\}>

  - Write (\{7\}, A)
  - Write (\{0,1,2\}, BCD)
  - Write (\{5\}, E)
  - Write (\{3\}, F)
  - Write (\{11\}, G)
  - Write (\{15\}, H)
  - Write (\{0\}, I)
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
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
### 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
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
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
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**

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

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
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
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%

(a) PIC trace  (b) MP3 trace  (c) MOV trace  (d) PMP trace
(e) PC trace  (f) PCMark trace  (g) Install trace  (h) SYSmark trace
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>Data blocks</th>
<th>BAST</th>
<th>FAST</th>
<th>Superblock</th>
<th>Page mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminology</td>
<td>Data blocks</td>
<td>Data blocks</td>
<td>D-blocks</td>
<td>Data 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>Max degree of sharing</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</th>
<th>Log blocks</th>
<th>SW / RW log blocks</th>
<th>U-blocks</th>
<th>Update blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max degree of sharing</td>
<td>1</td>
<td>1 (sequential) or P (random)</td>
<td>Min(P, S)</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block merge</th>
<th>Frequency</th>
<th>Middle</th>
<th>Low</th>
<th>Low</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Cost</td>
<td>Middle</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>