Flash Storage

Operating Systems
Based on: Three Easy Pieces by Arpaci-Dusseaux

Carmi Merimovich

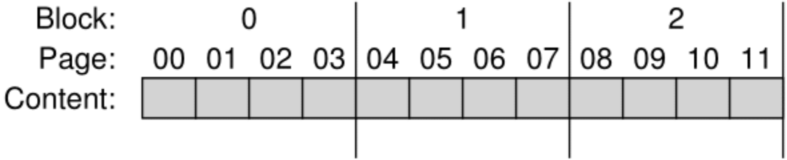
Tel-Aviv Academic College

Stroing Bits

- Floating transistors. (Basic element of EEPROM)
- In EEPROM the whole chip needs to be erased.
- In Flash memory erasures can be partial.
- SLC: Single Level Cell.
- MLC: Multiple Level Cell.
- TLC: Triple Level Cell.
- SLC achieves higher performance. (and more expensive)

Bits to Banks/Planes

- Bank/Plane contains many cells.
- Banks are accessed using two size units:
  1. (erase) blocks: About 128KB – 4MB.
  2. pages: About 2KB, 4KB.
- 3 blocks each of 4 pages:



Basic Flash Operations

- Read page: Give page number. 10s of μs.
- Erase block: Give block number. All cells in it will be 1. ms range.
- Program block: Write to an erased block. 100s of μs.

Device	Read (μs)	Program (μs)	Erase (μs)
SLC	25	200-300	1500-2000
MLC	50	600-900	~3000
TLC	~75	~900-1350	~4500

Figure 44.2: **Raw Flash Performance Characteristics**

Writing

		`iiii`	*Initial: pages in block are invalid* (`i`)
Erase()	$\rightarrow$	`EEEE`	*State of pages in block set to erased* (`E`)
Program(0)	$\rightarrow$	`VEEE`	*Program page 0; state set to valid* (`V`)
Program(0)	$\rightarrow$	**error**	*Cannot re-program page after programming*
Program(1)	$\rightarrow$	`VVEE`	*Program page 1*
Erase()	$\rightarrow$	`EEEE`	*Contents erased; all pages programmable*

Write detailed example

- We wish to write to page 0:

Page 0	Page 1	Page 2	Page 3
00011000	11001110	00000001	00111111
VALID	VALID	VALID	VALID

Page 0	Page 1	Page 2	Page 3
11111111	11111111	11111111	11111111
ERASED	ERASED	ERASED	ERASED

Page 0	Page 1	Page 2	Page 3
00000011	11111111	11111111	11111111
VALID	ERASED	ERASED	ERASED

- We loose pages 1-3!
- wear-out: Number of writes is not inifinite.

Relaibility

- Main problem: Wear-out.
  - Exact live-span unknown.
  - SLC: 100,000 Erase/Program cycles.
  - MLC: 10,000 Erase/Program cycles.
  - Probably much more.
- Disturbance: Read/Program from/to page changes near by cells.

Disk?

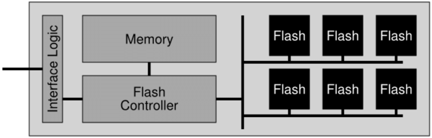


Figure 44.3: **A Flash-based SSD: Logical Diagram**

- Build something which looks like a 'normal' disk.
- FTL - Flash Translation Layer
  1. Accept read/write commands on logical number.
  2. Translate to flash operations.
- Should have good performance:
  1. Parallel flash chips.
  2. Reduce write amplifications.
  3. Should be reliable: Mainly wear leveling.

FTL - Bad Approach

Direct map.

- Read to logical page N translate to read physical page N.
- Write to logical page N translate to:
  1. Read block containing physical page N.
  2. Erase this block.
  3. Write the block with the modified page.

Note

- Write is costly.
- Write amplification.
- Wearing-out.

FTL - Log Structured

- We studied Log structures FS.
- Example:
  1. write(100) with a1
  2. write(101) with a2
  3. write(2000) with b1
  4. write(2001) with b2

Example - initial state

Block:	0				1				2			
Page:	00	01	02	03	04	05	06	07	08	09	10	11
Content:												
State:	i	i	i	i	i	i	i	i	i	i	i	i

Example: write (100)

Write(100) will translate to write to physical block 0.

1. Erase.

Block:	0				1				2			
Page:	00	01	02	03	04	05	06	07	08	09	10	11
Content:												
State:	E	E	E	E	i	i	i	i	i	i	i	i

2. Program.

Block:	0				1				2			
Page:	00	01	02	03	04	05	06	07	08	09	10	11
Content:	a1											
State:	V	E	E	E	i	i	i	i	i	i	i	i

3. Record location

Table: 100 → 0 — Memory

Flash Chip

Block:	0				1				2			
Page:	00	01	02	03	04	05	06	07	08	09	10	11
Content:	a1											
State:	V	E	E	E	i	i	i	i	i	i	i	i

Example: After all four writes

Table: 100 → 0 101 → 1 2000 → 2 2001 → 3 Memory

Block:	0				1				2				Flash Chip
Page:	00	01	02	03	04	05	06	07	08	09	10	11	
Content:	a1	a2	b1	b2									
State:	V	V	V	V	i	i	i	i	i	i	i	i	

Note

- Erases are done once a while.
- No Read-Modify-Write.
- Wear leveling is automatic.

Log FTL issues

- Need to save the mapping table.
  1. Reserve area in each page for part of table. Large disk reconstruction might be long.
  2. High end devices use a form of logging and checkpointing.
- Garbage collection.
- Note. Excessvie garabage collection causes write amplification.
- Mapping tables might be huge.

Garbage collection

1. write (100) c1
2. write (100) c2

Table: 100 → 4 101 → 5 2000 → 2 2001 → 3 Memory

Block:	0				1				2				
Page:	00	01	02	03	04	05	06	07	08	09	10	11	Flash
Content:	a1	a2	b1	b2	c1	c2							Chip
State:	V	V	V	V	V	V	E	E	i	i	i	i	

- The same garbage collection from log-fs is used.
- Identifying dead block also uses the methof of log-fs. (which is?)

State after garbage collection

Table: 100 → 4 101 → 5 2000 → 6 2001 → 7 Memory

Block:	0				1				2				
Page:	00	01	02	03	04	05	06	07	08	09	10	11	Flash
Content:					c1	c2	b1	b2					Chip
State:	E	E	E	E	V	V	V	V	i	i	i	i	

- Expensive
- Best if all blocks are dead. Then only Erase is needed.

Reduce mapping Table Size (1)

Use block mapping instead of page mapping.

1. Write (2000)
2. Write (2001)
3. Write (2002)
4. Write (2003)

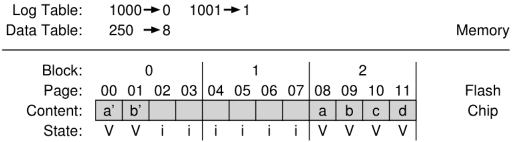
Table: 500 → 4 Memory

Block:	0				1				2				Flash Chip
Page:	00	01	02	03	04	05	06	07	08	09	10	11	
Content:					a	b	c	d					
State:	i	i	i	i	V	V	V	V	i	i	i	i	

Reduce mapping Table Size (2)

Use block mapping instead of page mapping.

1. Write (2003)



- Smaller tables.
- Writing a logical block is expensive.
- (Physical block is 256KB or some such)

Hybrid Mapping

- Several blocks are considered log blocks.
- Writing to them is done as usual.
- There is a mapping to pages inside these blocks.
- This is the log table.
- There is also a data table which maps blocks.
- Read logical blocks search first in log table and then in data table.

Hybrid Mapping key point

- Number of log blocks should be small.
- Thus log blocks should be switched to data blocks at some point.

Switch Merge (1)

1. Begining state:

Log Table:
Data Table: 250 → 8 — Memory

Block:	0				1				2				
Page:	00	01	02	03	04	05	06	07	08	09	10	11	Flash
Content:									a	b	c	d	Chip
State:	i	i	i	i	i	i	i	i	V	V	V	V	

2. Overwriting logical blocks 1000–1003 with a’,b’,c’,d’

Log Table: 1000 → 0 1001 → 1 1002 → 2 1003 → 3
Data Table: 250 → 8 — Memory

Block:	0				1				2				
Page:	00	01	02	03	04	05	06	07	08	09	10	11	Flash
Content:	a’	b’	c’	d’					a	b	c	d	Chip
State:	V	V	V	V	i	i	i	i	V	V	V	V	

Switch Merge (2)

Log Table:

Data Table: 250 → 0 Memory

Block:	0				1				2				
Page:	00	01	02	03	04	05	06	07	08	09	10	11	Flash
Content:	a’	b’	c’	d’									Chip
State:	V	V	V	V	i	i	i	i	i	i	i	i	

Partial Merge (1)

1. Begining state:

Log Table:
Data Table: 250 → 8

Memory

Block:	0				1				2			
Page:	00	01	02	03	04	05	06	07	08	09	10	11
Content:									a	b	c	d
State:	i	i	i	i	i	i	i	i	V	V	V	V

Flash Chip

2. Overwriting logical blocks 1000,1001 with a',b'.

Log Table: 1000 → 0 1001 → 1
Data Table: 250 → 8

Memory

Block:	0				1				2			
Page:	00	01	02	03	04	05	06	07	08	09	10	11
Content:	a'	b'							a	b	c	d
State:	V	V	i	i	i	i	i	i	V	V	V	V

Flash Chip

Partial Merge (2)

1. Additional read is needed to complete the switch.

Log Table:
Data Table: 250 → 0 — Memory

Block:	0				1				2				
Page:	00	01	02	03	04	05	06	07	08	09	10	11	Flash
Content:	a’	b’	c’	d’									Chip
State:	V	V	V	V	i	i	i	i	i	i	i	i	

Full Merge

1. Logical blocks 0,4,8,12 are written to log block A.
2. Switching requires:
    1. Read 0, 1, 2, 3, then writing a new data block.
    2. Read 4, 5, 6, 7, then writing a new data block.
    3. etc...

Mapping Table - Cache

Use a method with the same idea as in virtual memory.

- Store in memory mappings of actively used pages.
- Decide dynamically what is actively used pages.
- If there is enough memory for 'working set', very good performance.
- (The above means spatial and temporal locality.)
- If not enough memory, we are in trouble due to 'thrashing'.

Forced wear-leveling

- Blocks on flash which are not updating.
- Hence do not participate in the weal-leveling.
- Need to read also live blocks and write them elsewhere.

Performance

Device	Random Reads (MB/s)	Random Writes (MB/s)	Sequential Reads (MB/s)	Sequential Writes (MB/s)
Samsung 840 Pro SSD	103	287	421	384
Seagate 600 SSD	84	252	424	374
Intel SSD 335 SSD	39	222	344	354
Seagate Savvio 15K.3 HDD	2	2	223	223

Figure 44.4: **SSDs And Hard Drives: Performance Comparison**

- No mechanical parts.
- Much better in random read/writes.

Nowadays:

- 60 cents per GB for SSD.
- 5 cents per GB for HDD.