Data Storage

Operating Systems

Assoc. Prof. Milos Jovanovik, PhD

Stable Storage

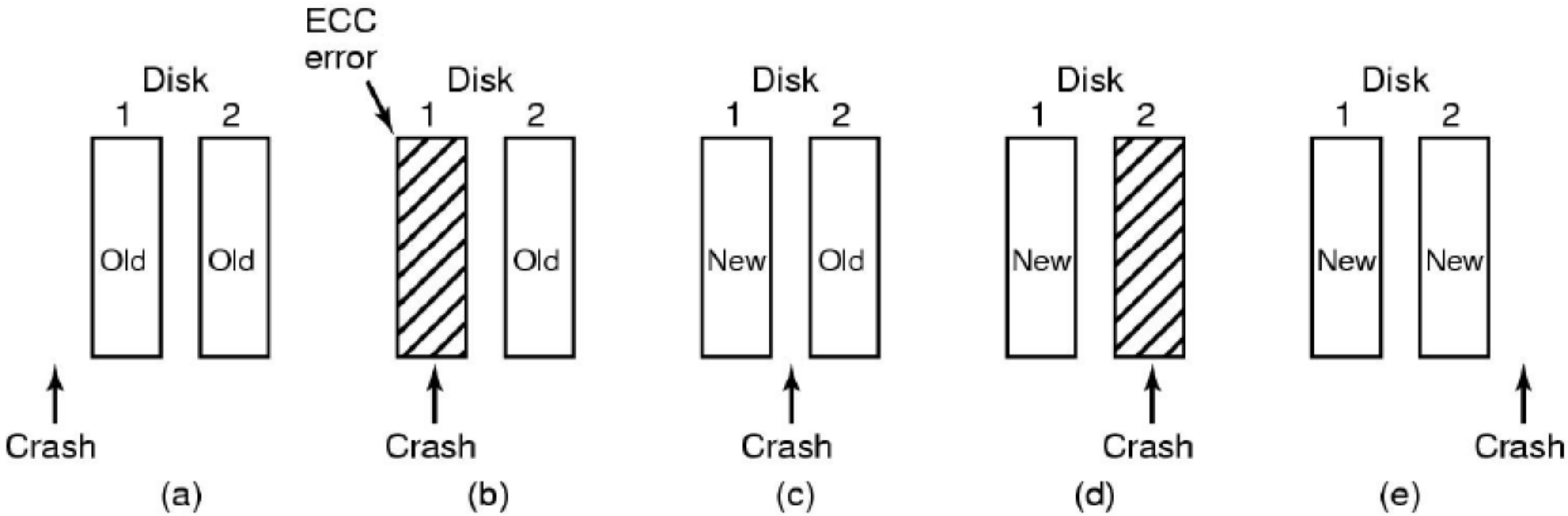
Stable Storage

- When a write is issued to the disk, it should either correctly write the data, or do nothing (leaving the existing data intact)
- Reasons:
  - ECC is not enough
  - A correctly written sector can spontaneously go bad and become unreadable
  - CPU can fail
- A pair of identical disks is used
- Implemented in software

Stable Storage

- Stable write
  - The data is written and verified (read) on both disks (first HD1, then HD2)
  - If the data is not verified, the operation is repeated up to **n** times
  - After n consecutive failures, the block is remapped onto a spare and the operation gets repeated until it succeeds
- Stable read
  - Reads the block from HD1 and checks ECC
  - If this yields an incorrect ECC, the read is tried again, up to **n** times.
  - If all of these give bad ECCs, the corresponding block is read from HD2
- Crash recovery
  - If a pair of blocks are both correct, nothing is done
  - If one of them has an ECC error, the bad block is overwritten with the corresponding good block
  - If a pair of blocks are both good but different, the block from HD1 is written onto HD2

Stable Storage



Analysis of the influence of crashes on stable writes.

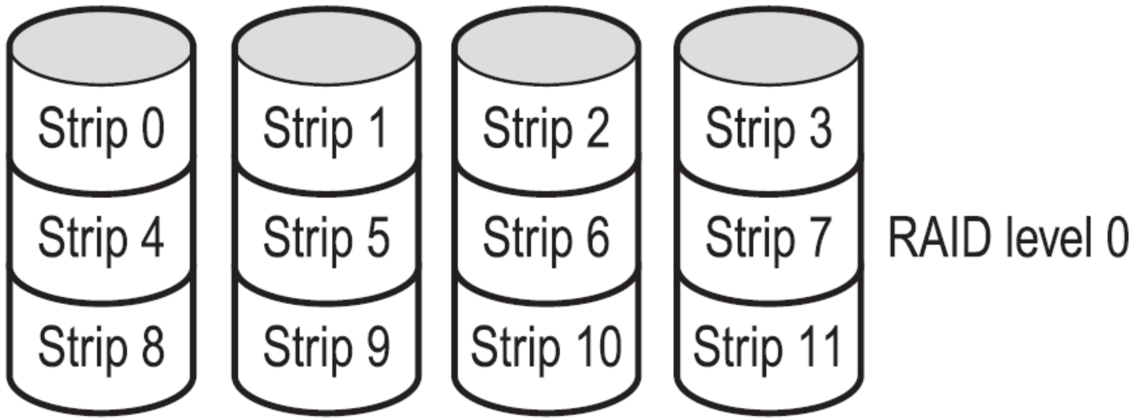
RAID

- In order to satisfy the demand for higher rates and parallel data processing, several different architectures are used for disk organization – RAID (Redundant Array of Inexpensive/Independent Disks).

- The basic concept is to connect several disks in order to achieve redundancy and availability.

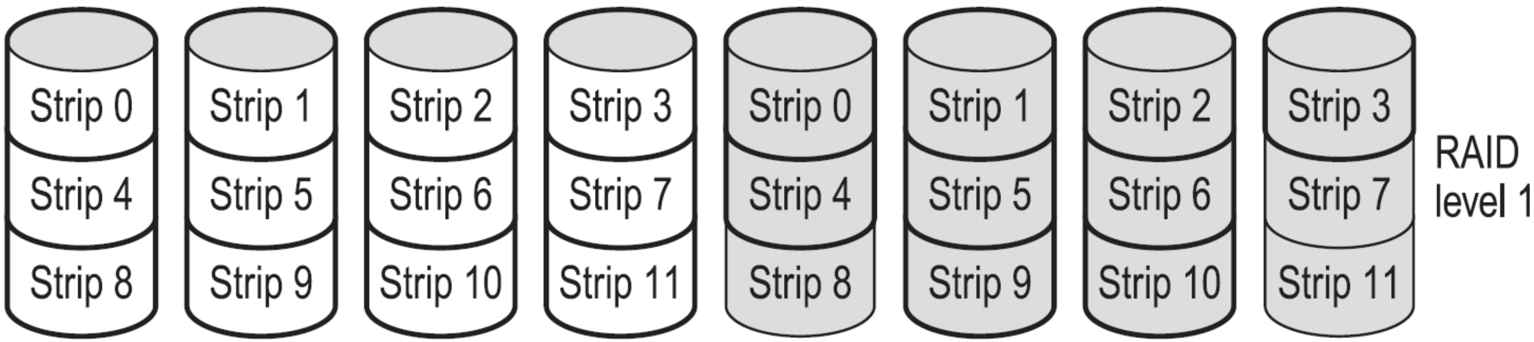
- Transparent to the processor.

RAID 0 – Strip Set



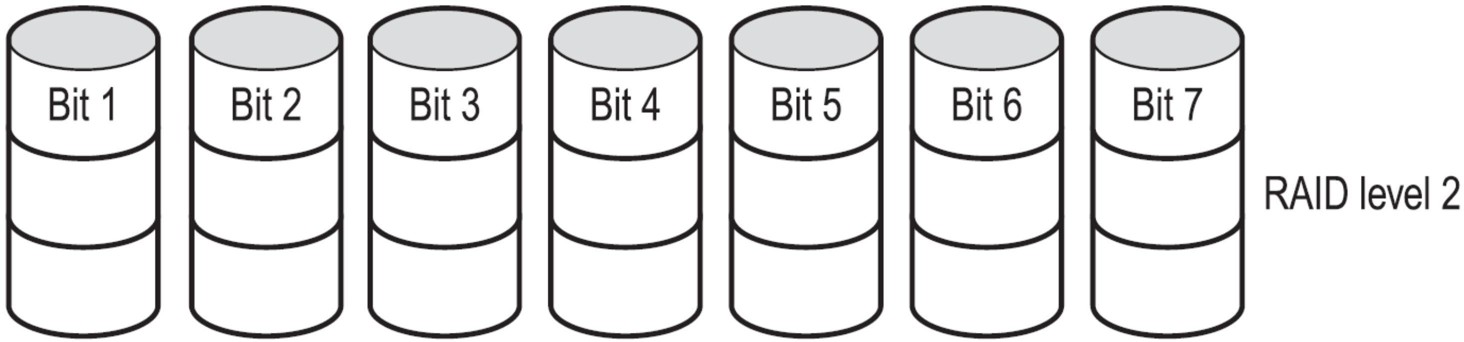
- Each strip contains k sectors.
- It is used to increase the performances – parallel I/O.
- The capacity depends on the smallest disk:
  - HD1 = 100GB, HD2 = 120 GB, Total = 200GB
- Negative parameter: MTTF = MTTF/#HD.

RAID 1



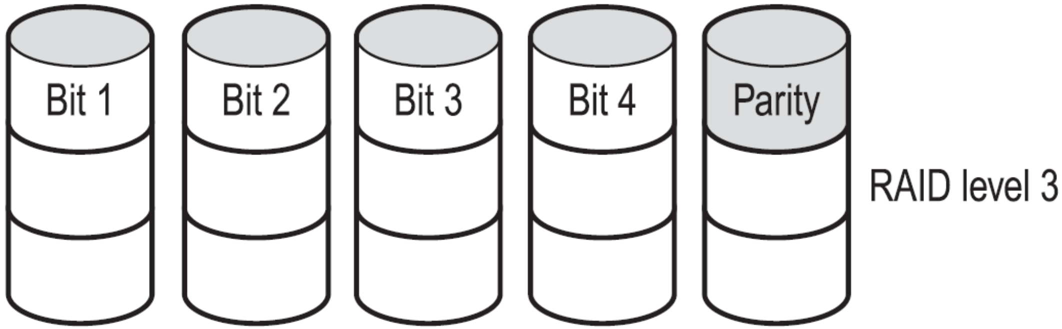
- It duplicates all the disks, so there are four primary disks and four backup disks.
- On a write, every strip is written twice (bad performance).
- On a read, either copy can be used, distributing the load over more drives (good performance).
- Fault tolerance is excellent.

RAID 2



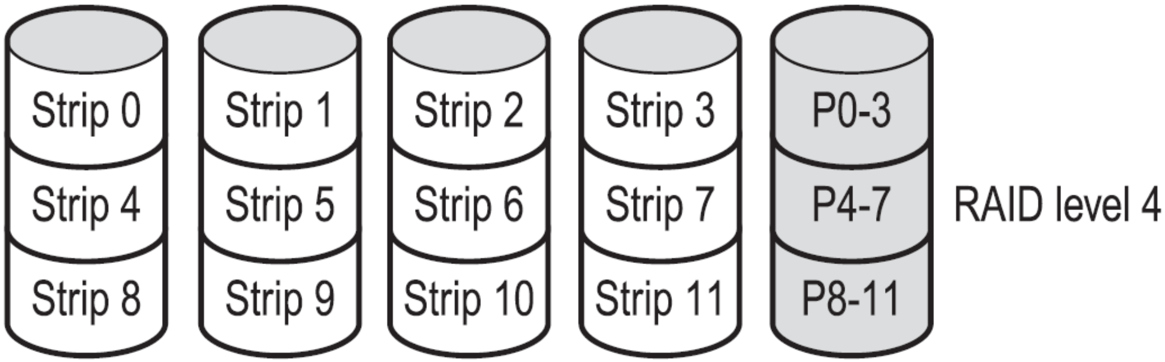
- Data is divided on a bit level and Hamming code is used for error correction.
- Possible high rates, but not used that often.
- We would need 39 disks (32 disks for a word, 7 for Hamming code and parity).
- This scheme requires all drives to be rotationally synchronized.

RAID 3



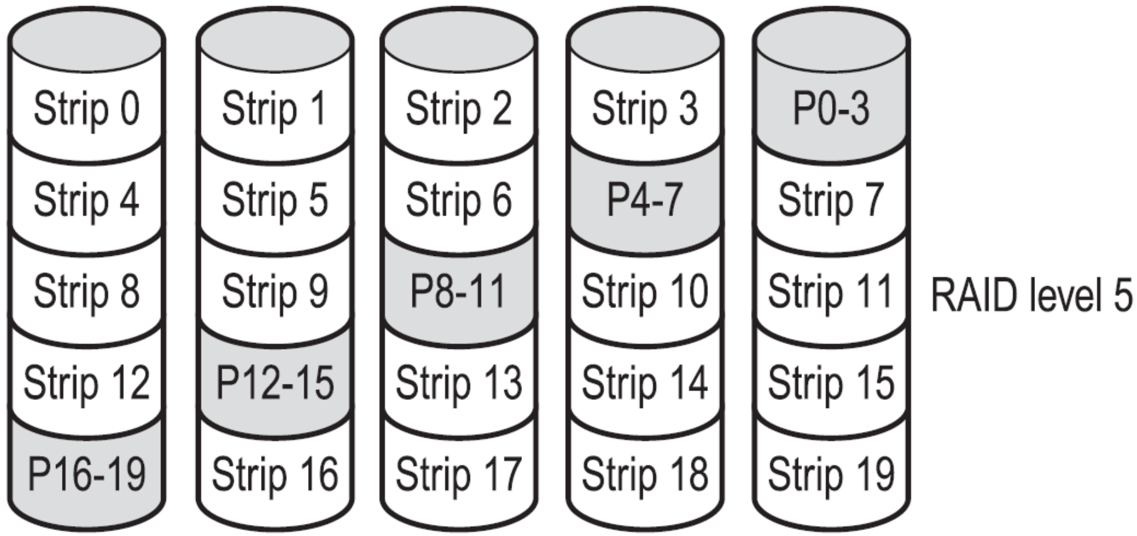
- RAID 3 consists of byte-level striping with dedicated parity.
- Although implementations exist, RAID 3 is not commonly used in practice.
- Error correction is possible
- The number of separate I/O requests per second they can handle is no better than for a single drive

RAID 4



- Strip-for-strip parity written onto an extra drive.
- It can serve several requests.
- The parity strip is obtained from the other 4 stripes with an XOR operation.
- If a drive crashes, the lost bytes can be recomputed from the parity drive by reading the entire set of drives.
- Negative side: When writing on one sector, all the disks needs to be read in order to update the corresponding parity stripe.

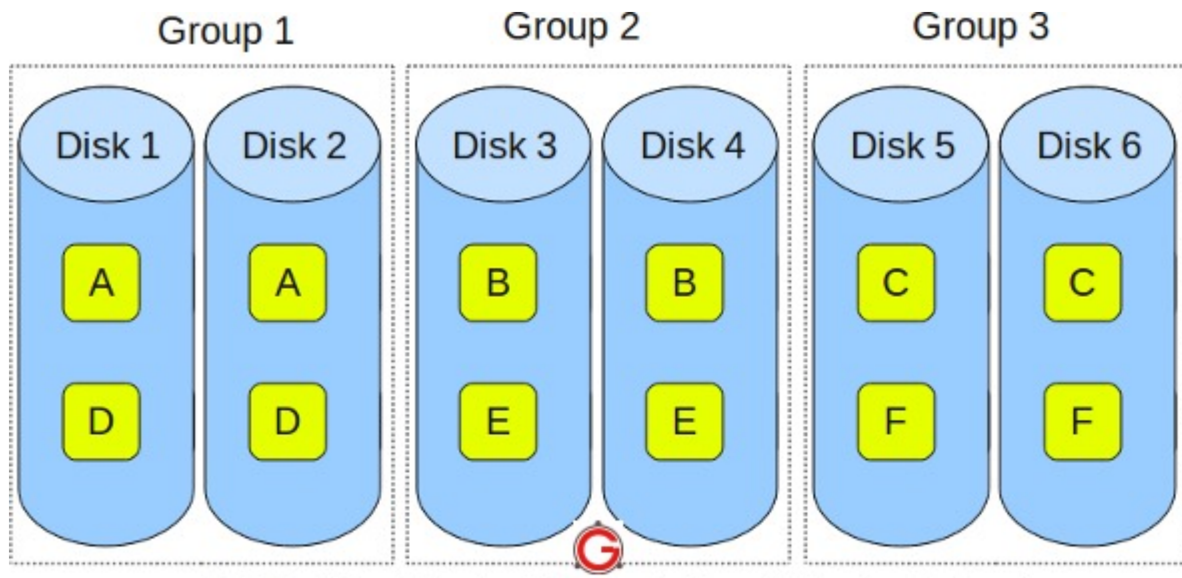
RAID 5



- As a consequence of the heavy load on the parity drive, it may become a bottleneck.
- This bottleneck is eliminated in RAID level 5 by distributing the parity bits uniformly over all the drives, in a round-robin fashion.
- However, in the event of a drive crash, reconstructing the contents of the failed drive is a complex process.

RAID 10

RAID 10 == RAID 1+0

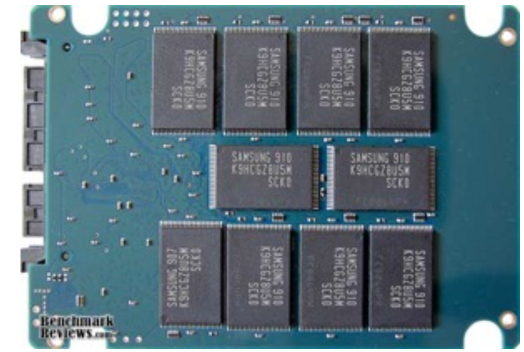


RAID 10 – Blocks Mirrored. (and Blocks Striped)

- It needs at least 4 disks
- The disks are grouped in pairs as mirrored disks
- For 6 disks in RAID 10, there are 3 groups
- In one group, the data are mirrored – Disk 1 and Disk 2 are in group 1
- In the group, the data is stretched, i.e. Block A is written in group 1, Block B in Group 2, Block C in group 3.
- It is called "stripe of mirrors", i.e. the disks in the group are mirrored and the data spans across groups.

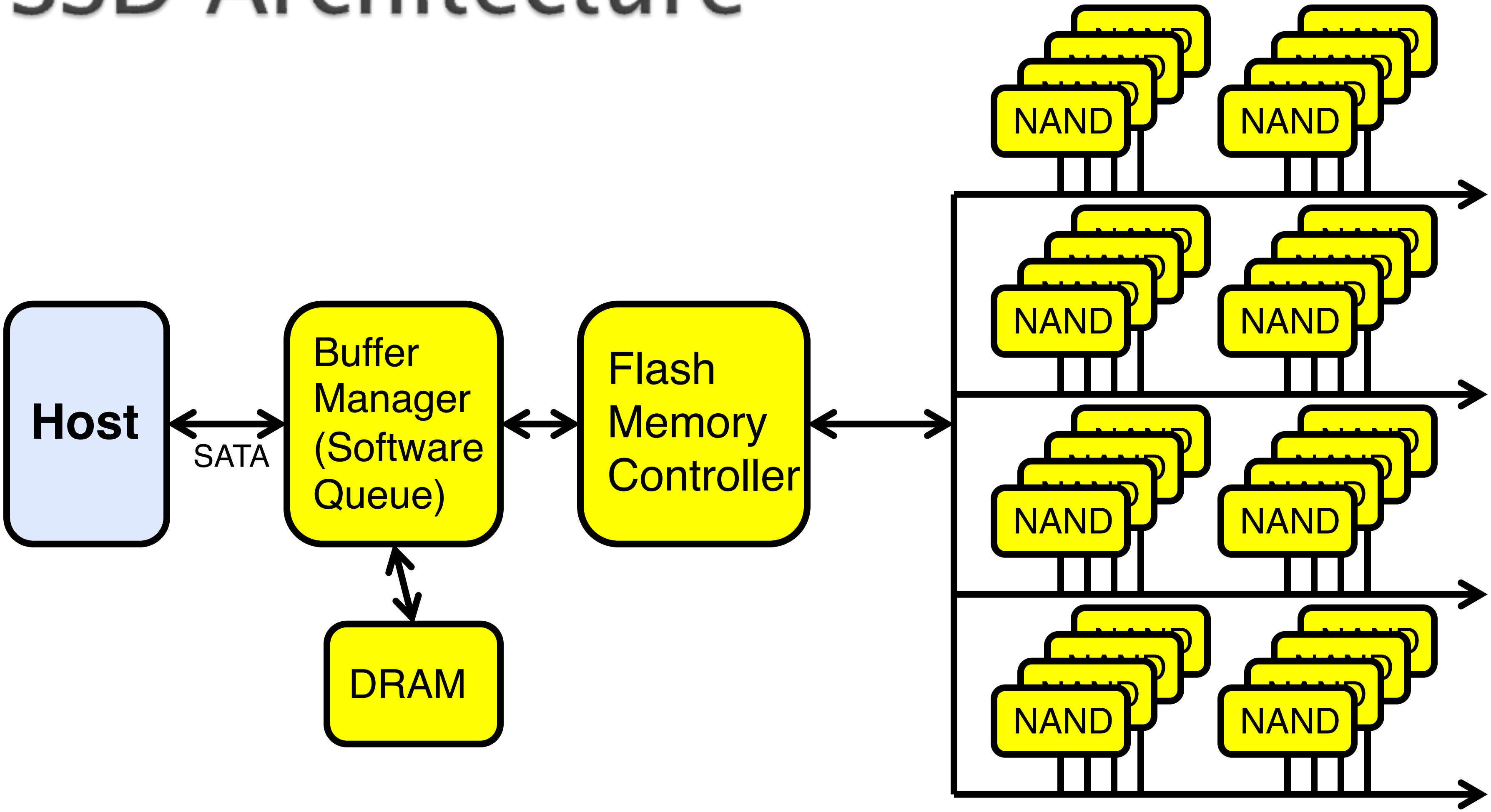
Solid State Drives (SSDs)

Solid State Disks (SSDs)



- 1995 – Replace rotating magnetic media with non-volatile memory (battery backed DRAM)
  - 2009 – Use NAND Flash: Single-Level Cell (1-bit/cell), Multi-Level Cell (2-bit/cell)
- Sector (4 KB page) addressable, but stores 4-64 "pages" per memory block
- No moving parts (no rotate / seek motors)
  - Eliminates seek and rotational delay (0.1-0.2ms access time)
  - Very low-power and lightweight

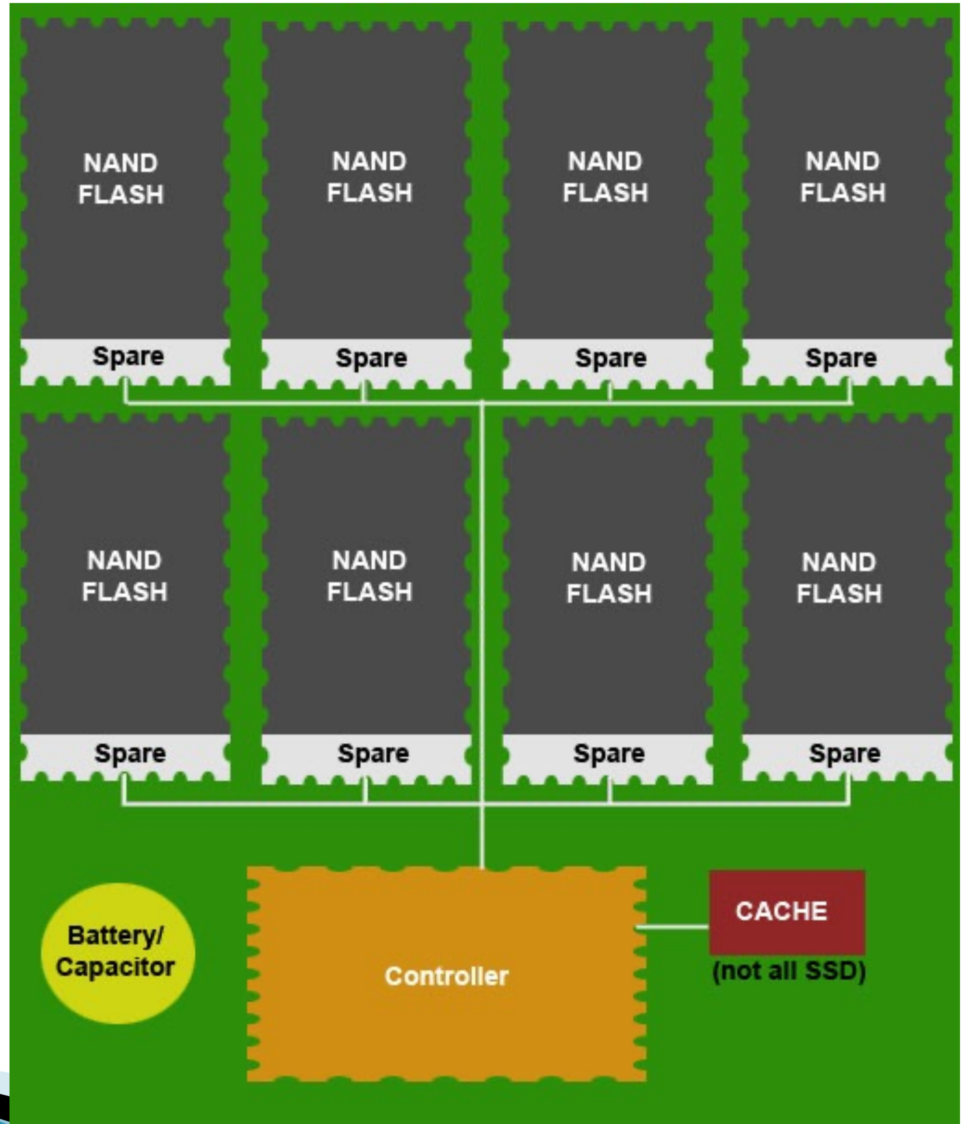
SSD Architecture



SSD Architecture

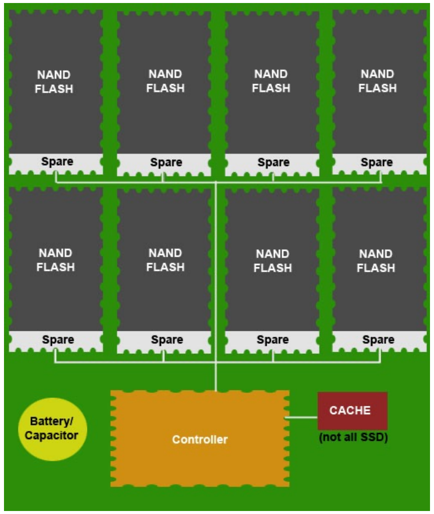
- SSD is divided into **blocks**, in which **pages** are kept.
- Pages consist of adjacent cells from NAND Flash memory.
- Blocks are grouping pages, and the number of blocks limits the size of the SSD.
- Pages are 4KB and blocks have 64 pages.
- **Data is written page-by-page, but deleted only block-by-block.**
  - Data is erased from 256KB (64*4KB) blocks.

SSD Architecture Example



SSD Architecture Example

- 128 GB SSD with 8 NAND Flash chips
- Each chip is 16 GB or total of 120 GB, because 1GB is a spare chip
- The cache is not always present, and it has directories of which blocks are occupied and wear leveling
- Usually there are 4 to 10 channels to the NAND chips, which increases the throughput.
- Battery / Capacitor is used when there are problems with the power supply, in order to finish the process of writing on the SSD

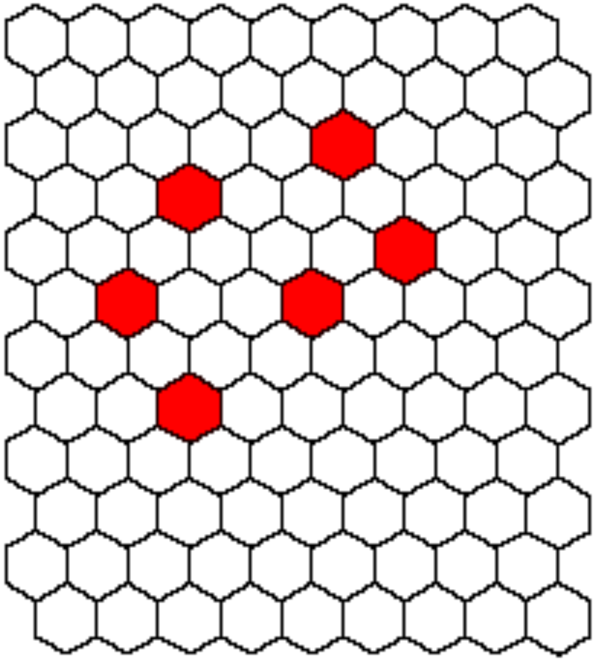


SSD: Read

- Reading data similar to memory read (25μs)
- No seek or rotational latency
- Transfer time: transfer a 4KB page
    - Limited by controller and disk interface (SATA: 300–600MB/s)
- Latency = Queuing Time + Controller time + Transfer Time
- Highest Bandwidth: Sequential OR Random reads

Writing on SSD

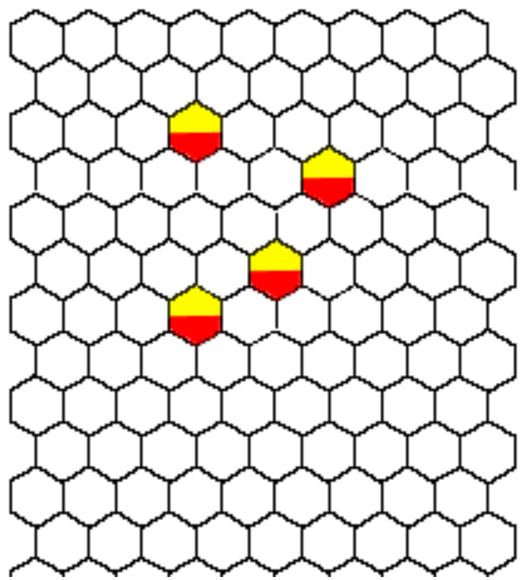
- For SLC (single level cell), the cell is either On or Off.



- Writing is easier, there is less work for the controller

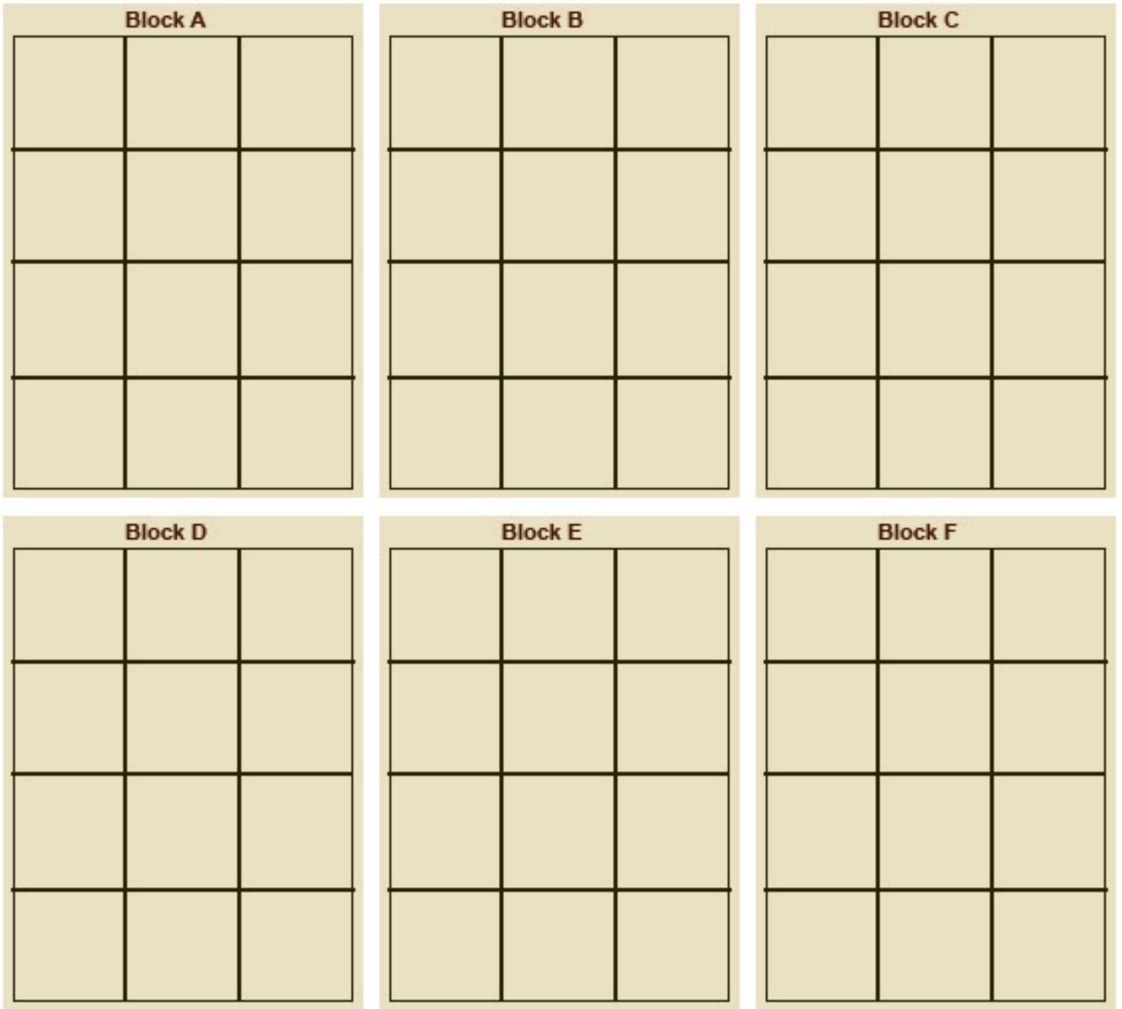
Writing on SSD

- Multi-level cell, the cell is On On, On Off, Off On or Off Off.
- It doubles the capacity
- It compromises reliability



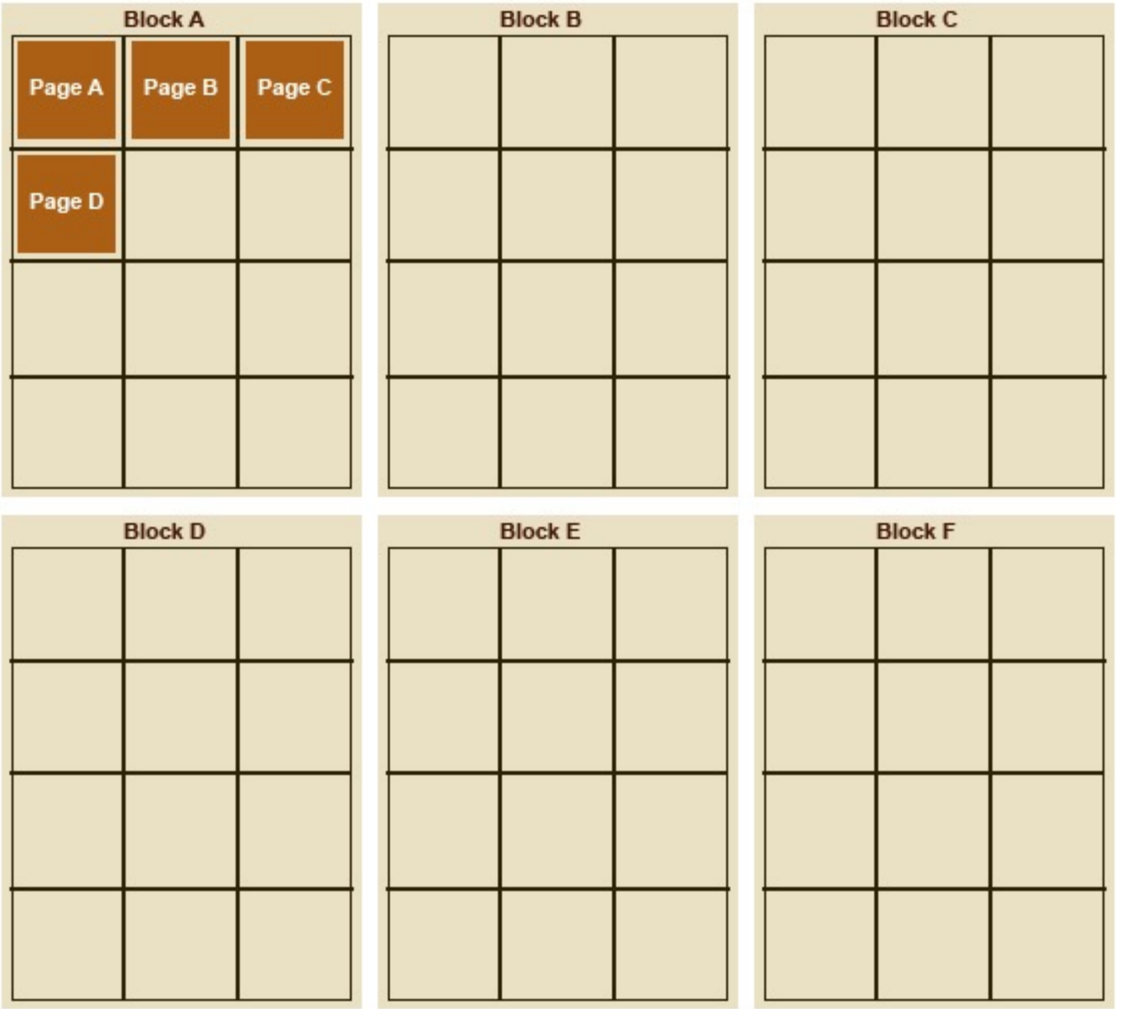
Writing on SSD

- For instance, if the block size is 12 pages, each with size of 1B



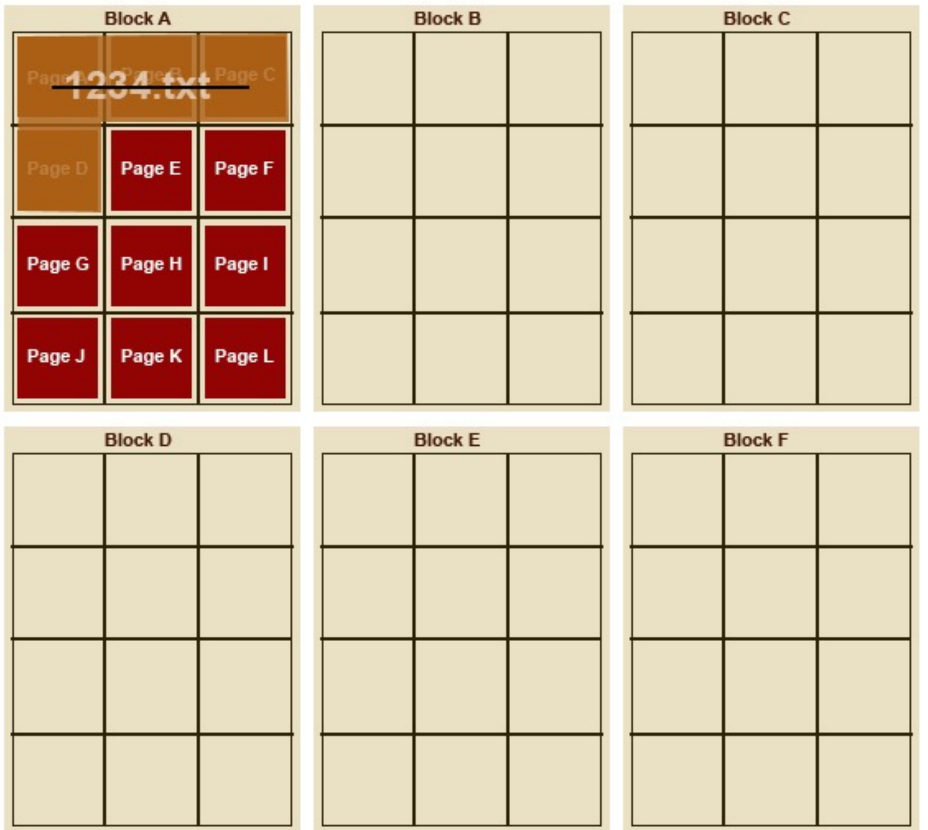
Writing on SSD

- Let us assume a file of 4B -> 4 pages



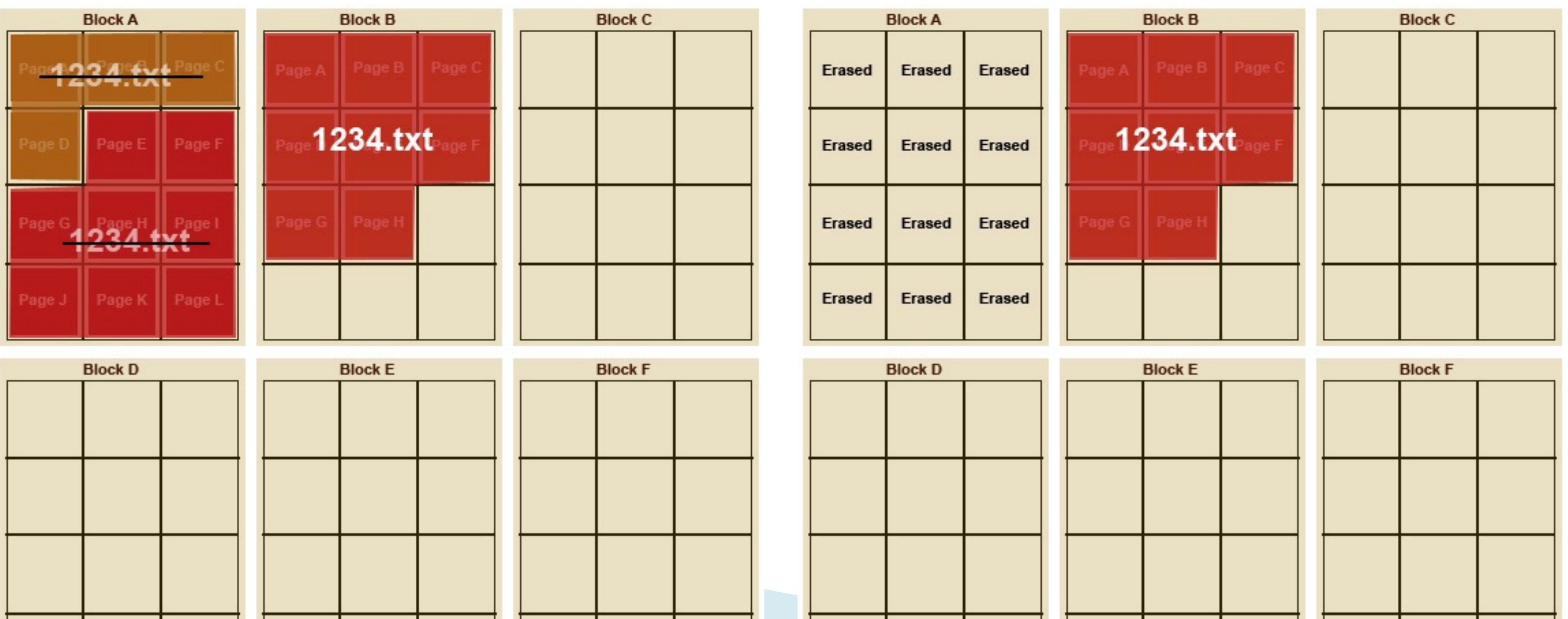
Writing on SSD

- If we change the file and we add text with size of 8B.
- Old pages are marked for deleting, whereas the new pages are written in the block



Writing on SSD

- SSD erases only full blocks, not pages!
- In order to erase old data, the new data must be written on block B, in order to erase data from block A

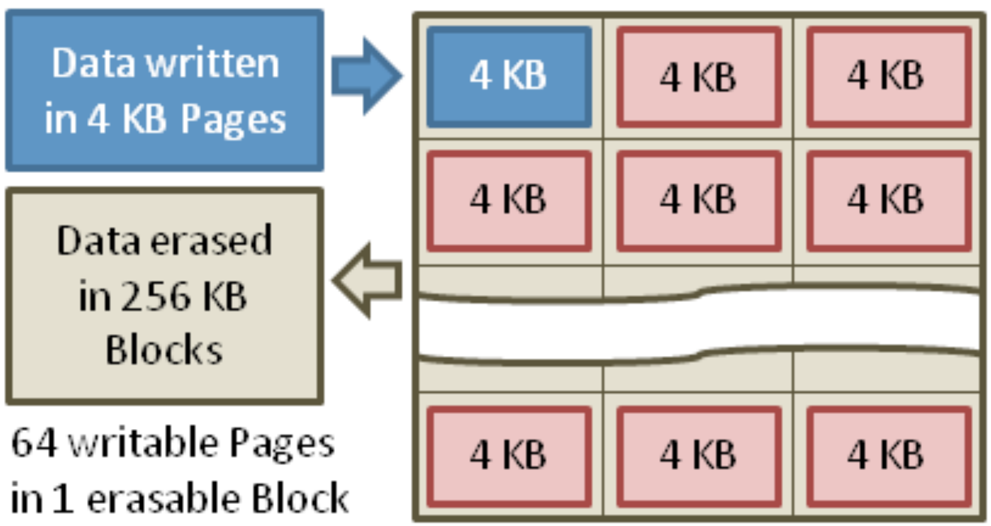


Writing on SSD

- What happens if we delete the file xyz.dll?

Writing on SSD

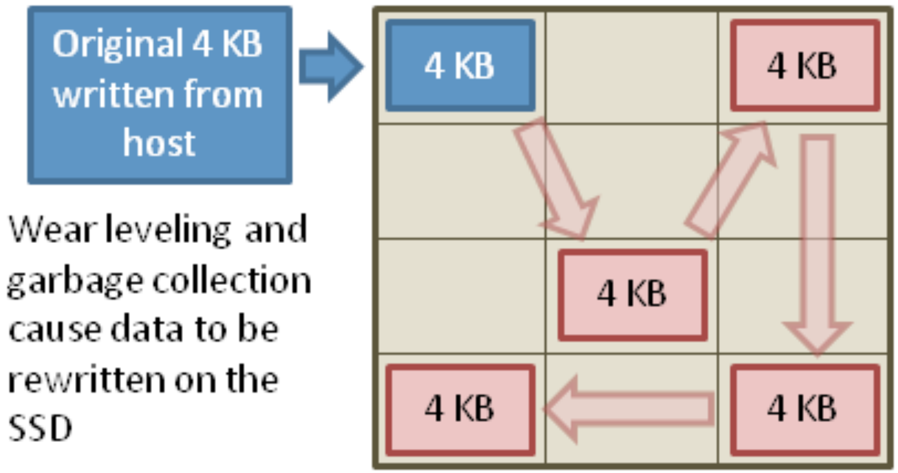
- Writing data is complex! (~200µs – 1.7ms)
  - Only empty pages can be written in a block
  - Only entire block can be erased (~1.5ms)
  - Controller maintains pool of empty blocks by coalescing used pages (read, erase, write)
    - reserves some % of capacity



Typical NAND Flash Pages and Blocks

Writing on SSD

- Write and erase cycles require “high” voltage
  - Damages memory cells, limits SSD lifespan
  - Controller uses ECC, performs wear leveling



- Result is very workload-dependent performance
  - Latency = Queuing Time + Controller time (Find Free Block) + Transfer Time
  - Highest BW: Seq. OR Random writes (limited by empty pages)

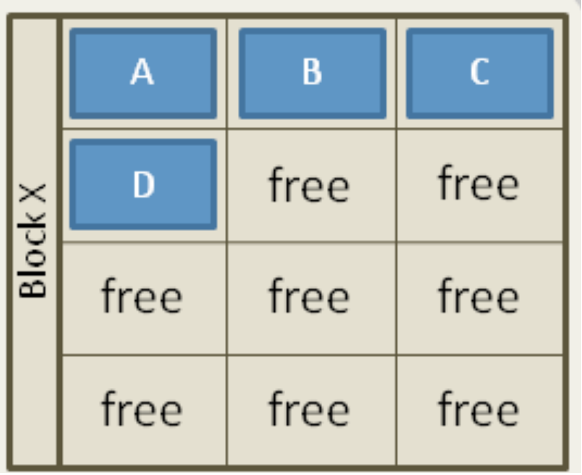
Rule of thumb: writes 10x more expensive than reads, and erases 10x more expensive than writes

Writing on SSD

- Data cleaning is called **Garbage Collection** and keeps the SSD performances
- You never defragment SSD!
- The process of cleaning the SSD, before the data is written can be a downside, because it adds write activities which can slow the SSD's performance.
- You can write limited times on SSD

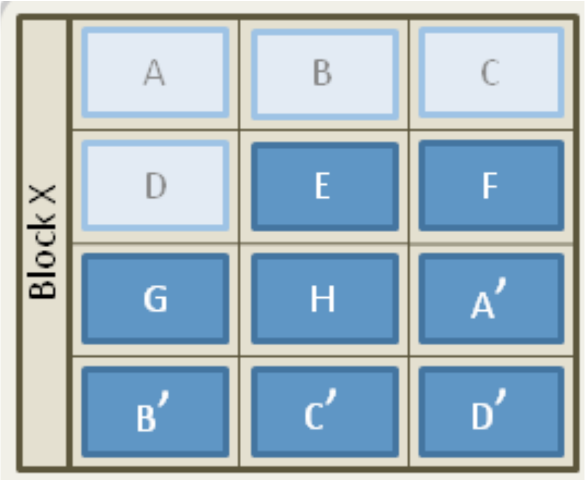
Writing on SSD

- Write A, B, C, D



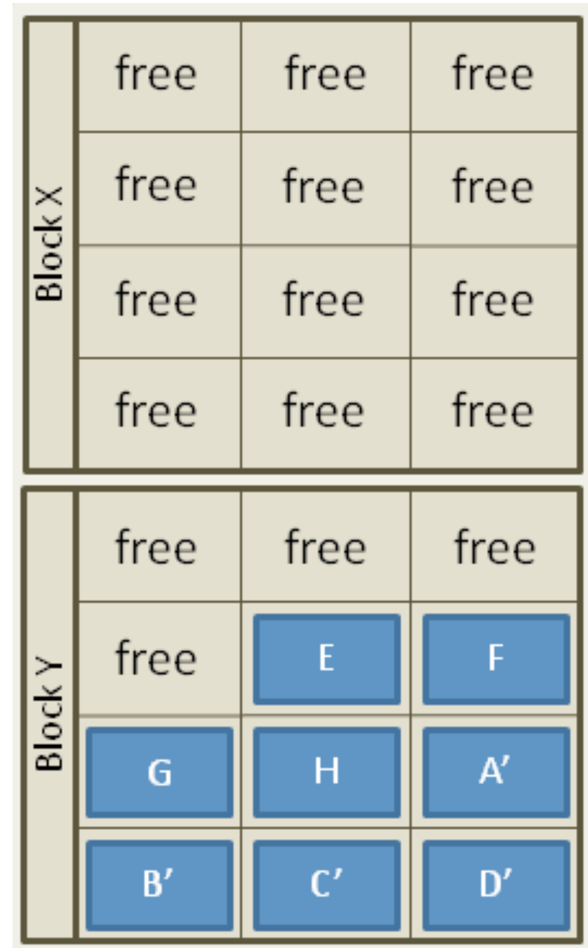
Writing on SSD

- Write A, B, C, D
- Write E, F, G, H and A’, B’, C’, D’
  - Record A, B, C, D as obsolete



Writing on SSD

- Controller *garbage collects* obsolete pages by copying valid pages to new (erased) block
- Typical steady state behavior when SSD is almost full
  - One erase every 64 or 128 writes



Storage Performance & Price

	Bandwidth (Sequential R/W)	Cost/GB	Size
HDD[2]	50-100 MB/s	$0.03-0.07/GB	2-4 TB
SSD[1,2]	200-550 MB/s (SATA) 6 GB/s (read PCI) 4.4 GB/s (write PCI)	$0.87-1.13/GB	200GB-1TB
DRAM[2]	10-16 GB/s	$4-14*/GB *SK Hynix 9/4/13 fire	64GB-256GB

[1] http://www.fastestssd.com/featured/ssd-rankings-the-fastest-solid-state-drives/

[2] http://www.extremetech.com/computing/164677-storage-pricewatch-hard-drive-and-ssd-prices-drop-making-for-a-good-time-to-buy

BW: SSD up to x10 than HDD, DRAM > x10 than SSD

Price: HDD x20 less than SSD, SSD x5 less than DRAM

SSD Summary

- Pros (vs. HDD):
  - Low latency, high throughput (eliminate seek/rotational delay)
  - No moving parts:
    - Very lightweight, low-power, silent, very shock insensitive
  - Read at memory speeds (limited by controller and I/O bus)

SSD Summary

- Cons
  - Small storage (0.1-0.5x HDD), very expensive (20x HDD)
    - Hybrid alternative: combine small SSD with large HDD
  - Asymmetric block write performance: read /erase/write
    - Controller garbage collection (GC) algorithms have major effect on performance
  - Limited drive lifetime
    - 1-10K writes/page for MLC NAND
    - Avg failure rate is 6 years, life expectancy is 9-11 years

Questions?