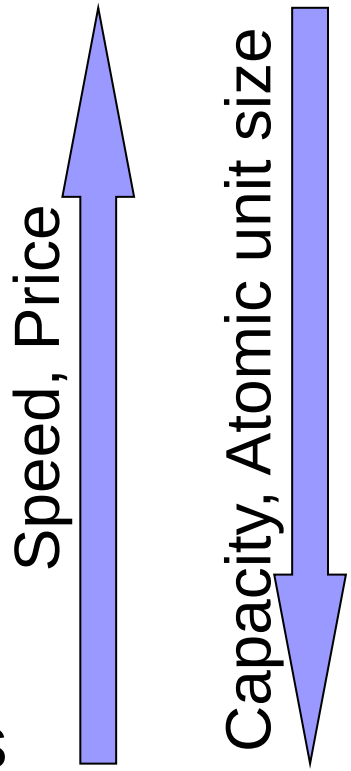
PA152: Efficient Use of DB

2. Data Storage

Vlastislav Dohnal

Storage Hierarchy

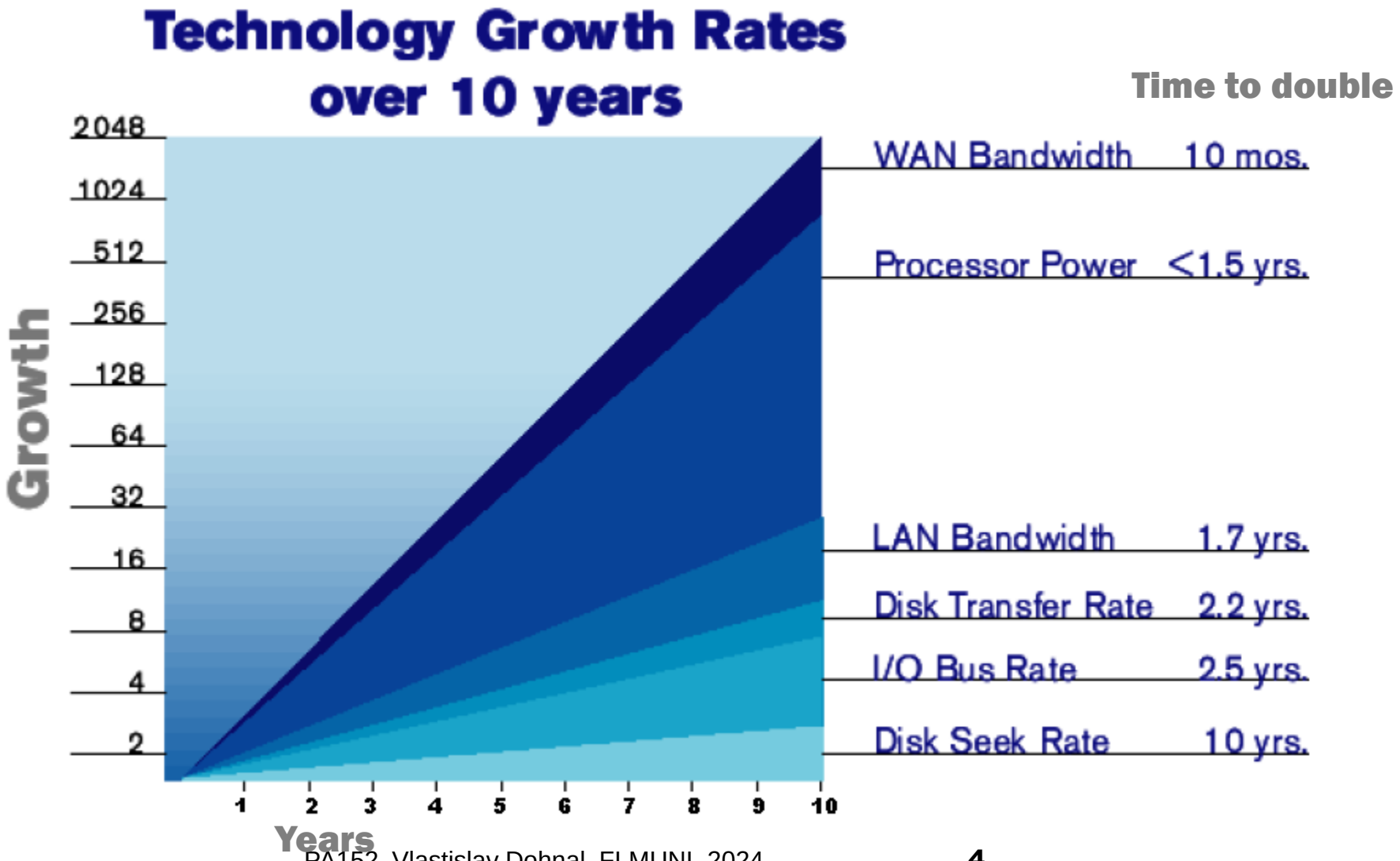
- Primary
  - cache
    - CPU
  - main (operation)
    - RAM
- Secondary
  - disk, flash
- Tertiary
  - tapes, optical discs
    - for backups



Empirical Laws

- Number of transistors
  - Density doubles each 2 years
  - CPU speed
  - Memory capacity
  - so-called Moore’s Law
- Disk capacity
  - Kryder's Law – 1000 times more in 15 years
  - Caveat: not applicable to disk speed!

Empirical Laws



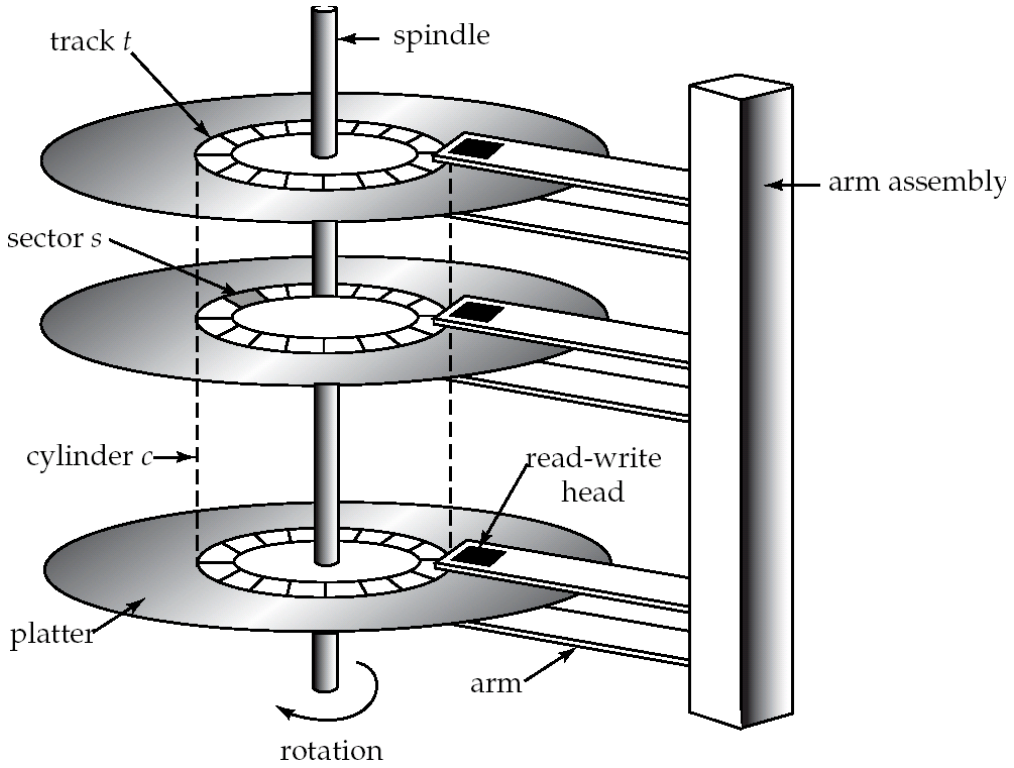
Storage Type

- Cache
  - Fastest, most expensive, volatile
- RAM
  - Fast – 10-100 ns (1 ns = 10^{-9} s)
  - Too (small or) expensive for the whole database
- Flash
  - Fast reads (25 µs = $25*10^{-6}$ s)
  - Slow writes (250 µs = $250*10^{-6}$ s)
  - Limited write cycles
- Magnetic disk
  - Sequential access fast (13.07ms + 100 ms = 113 ms)
  - Random access slow (13.07ms * 2560 = 33 459 ms)

Skipping details…

Magnetic disk

- Large capacity, non-volatile
- Reads and writes of the same speed (10ms)

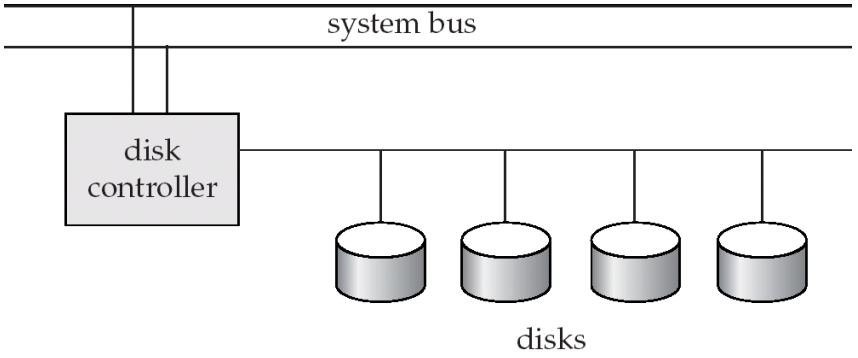


Magnetic disk (cont.)

- Access time
  - Time from a request for reading/writing to the beginning of data transfer
- Blocking data
  - disk sector/block is atomic unit
- Access time components
  - Seek time – 4-10 ms
    - Move heads to the correct track
    - Average seek time = ½ worst case seek time
  - Rotational delay – 4-11ms (5400-15k rpm)
    - Time for revolving to the correct sector
    - Average latency = ½ worse case latency

Magnetic disk (cont.)

- Transfer rate
  - Speed of reading/writing from/to a disk
  - 50-200MB/s, lower for inner tracks
- Speed of controller
  - More drives connected to one controller
  - SATA 3.2 (16Gb/s) – 2 GB/s
  - SAS-3 (12Gb/s) – 1.17 GB/s



Magnetic disk (cont.)

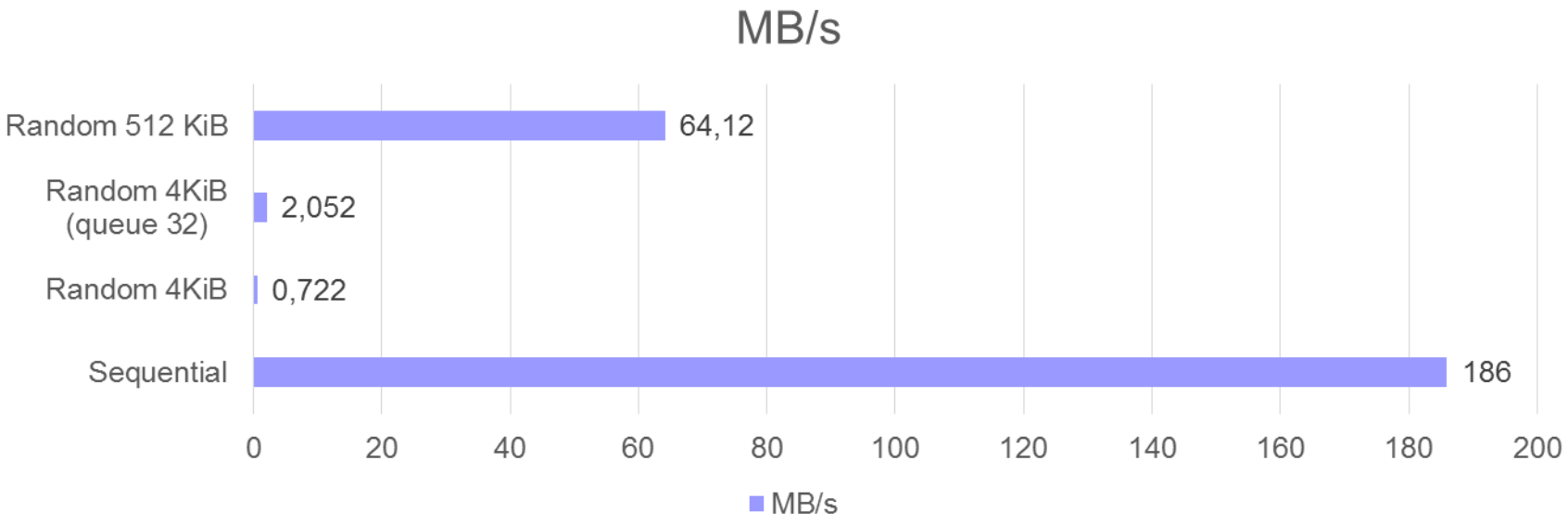
- Properties
  - Slow random read
    - access time can be up to 20ms
  - Fast sequential read
- Optimization in HW
  - Cache
    - Buffers for writing, battery backups or flash
  - Algorithms for arm moving minimization
    - Algorithm „elevator“
    - Work well when many concurrent requests

Magnetic disk (cont.)

- Example SATAII disk, 7200rpm, 100MB/s
  - Avg seek time = 8.9ms
  - Avg latency = (1/(7200/60))*0.5=0.00417s=4.17ms
  - Reading a block (4KB) = 13.07ms + 40μs
  - 10MB continuous = 13.07ms + 100ms = 113ms
    - consecutive block reading sometimes includes
      - track-to-track seek & platter / cylinder change
  - 10MB randomly = 2560*13.07488ms = 33.4s

Magnetic disk (cont.)

Western Digital 10EZEX 1TB, SATA3, 7200 RPM, sustained transfer rate 150 MB/s



Solid State Drives

- Disk storage on „flash“ memory
- No moving parts
- More resistant to damage
- Quiet, low access time and delay
- 4x more expensive than HDD (per GB)

SSD – Flash memory

- NAND chips
  - SLC (single-level cells)
    - stores 1-bit information (2 voltage states)
    - fastest, highest cost
  - MLC (multi-level cells)
    - stores mostly 2-bit information
  - TLC (triple-level cells)
    - stores 3-bit information (8 voltage states)
    - slowest, least cost
  - QLC (quad-level cells)

SSD – Flash performance

	SLC	MLC	TLC	HDD	RAM
P/E cycles	100k	10k	5k	*	*
Bits per cell	1	2	3	*	*
Seek latency (µs)	*	*	*	9000	*
Read latency (µs)	25	50	100	2000-7000	0.04-0.1
Write latency (µs)	250	900	1500	2000-7000	0.04-0.1
Erase latency (µs)	1500	3000	5000	*	*
Notes	* metric is not applicable for that type of memory				

Source: https://www.extremetech.com/extreme/210492-extremetech-explains-how-do-ssds-work

SSD – Flash memory

- Terminology shift! (block ➔ page)
- Organization
  - Pages of 4KiB organized into blocks (64/128 pages)
  - Much faster reading than writing
    - Write: Only in “a new block"
      1. write a new copy of the changed data to a fresh block
      2. remap the file pointers
      3. then erase the old block later when it has time
  - Write restrictions – 1k-100k overwrite cycles
  - Reliability increased ECC sums
    - Hamming, Reed-Solomon

SSD – Flash memory

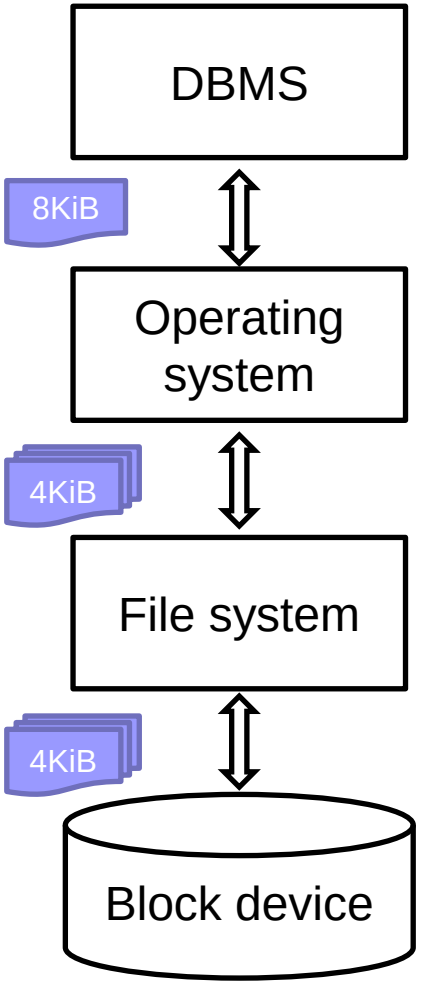
- A single NAND chip is relatively slow
  - SLC NAND
    - ~25 µs to read a 4 KiB page
    - ~250 µs to commit a 4 KiB page
    - ~2 ms to erase a 256 KiB block
- Data striping (RAID0) to improve performance

Disk ops ~ I/O operations

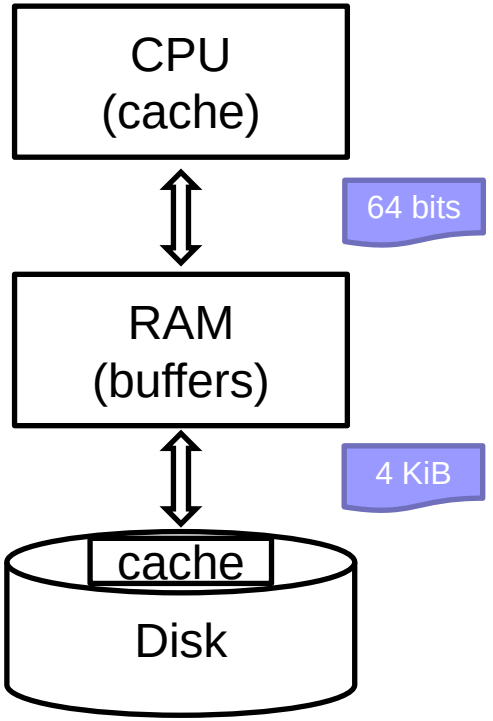
- Access in blocks (atomic unit)
  - Typically, 4KiB
- Block reading
- Block writing
  - Write and verify (rotate disk + reading!)
- Block update
  - Read
  - Change in memory
  - Write and verify

Data Exchange – Overview

SW level:



HW level:



Optimize Disk I/Os

- *Access Techniques*
  - *Minimize random access*
- Data Volume (item size)
  - Block size
- Storage Organization (IOPS)
  - Disk array

Techniques of Accessing Data

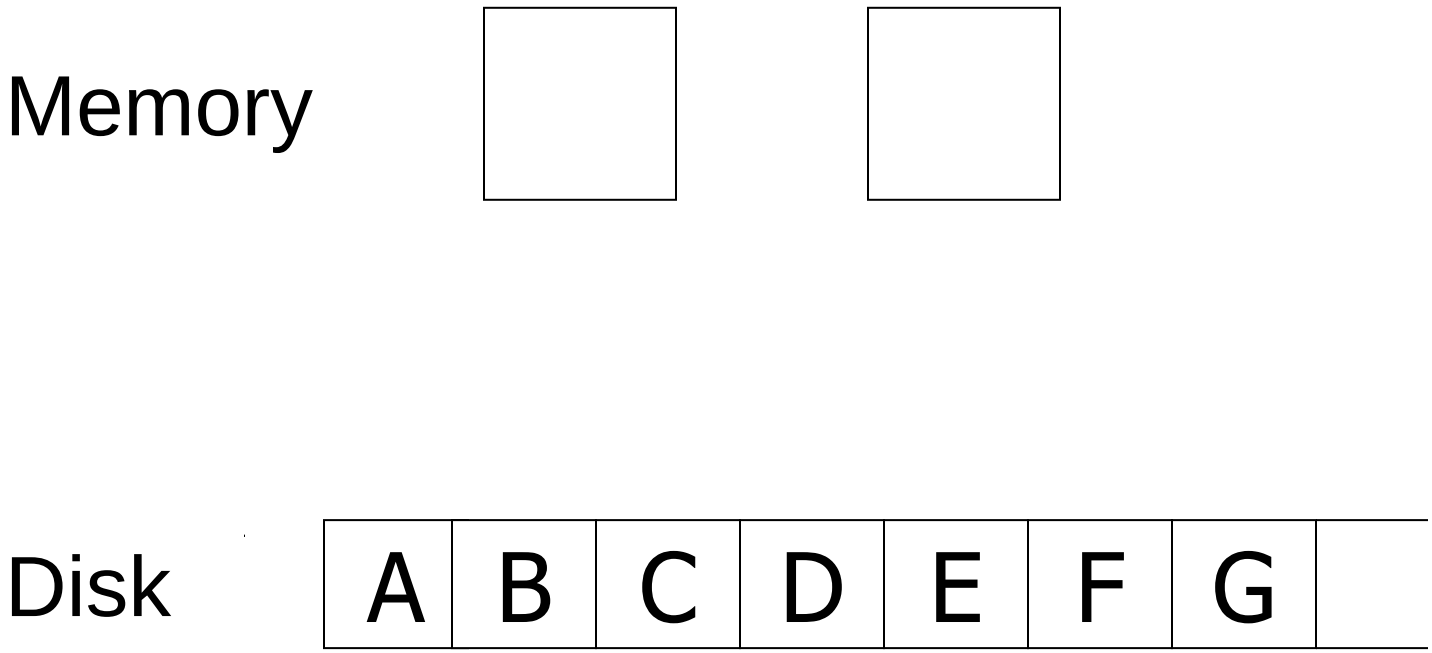
- App: Double buffering
- OS: Prefetching
- OS: Defragmentation
  - Arrange blocks in the order of processing
  - File system
    - Addressed at the file level
    - Allocate multiple blocks at once; disk defragmentation tool
- HW: Ordering access request (elevator algorithm)
  - Head movements in one direction
  - Re-order disk requests
    - Writes to battery-backup cache or log

Single Buffer

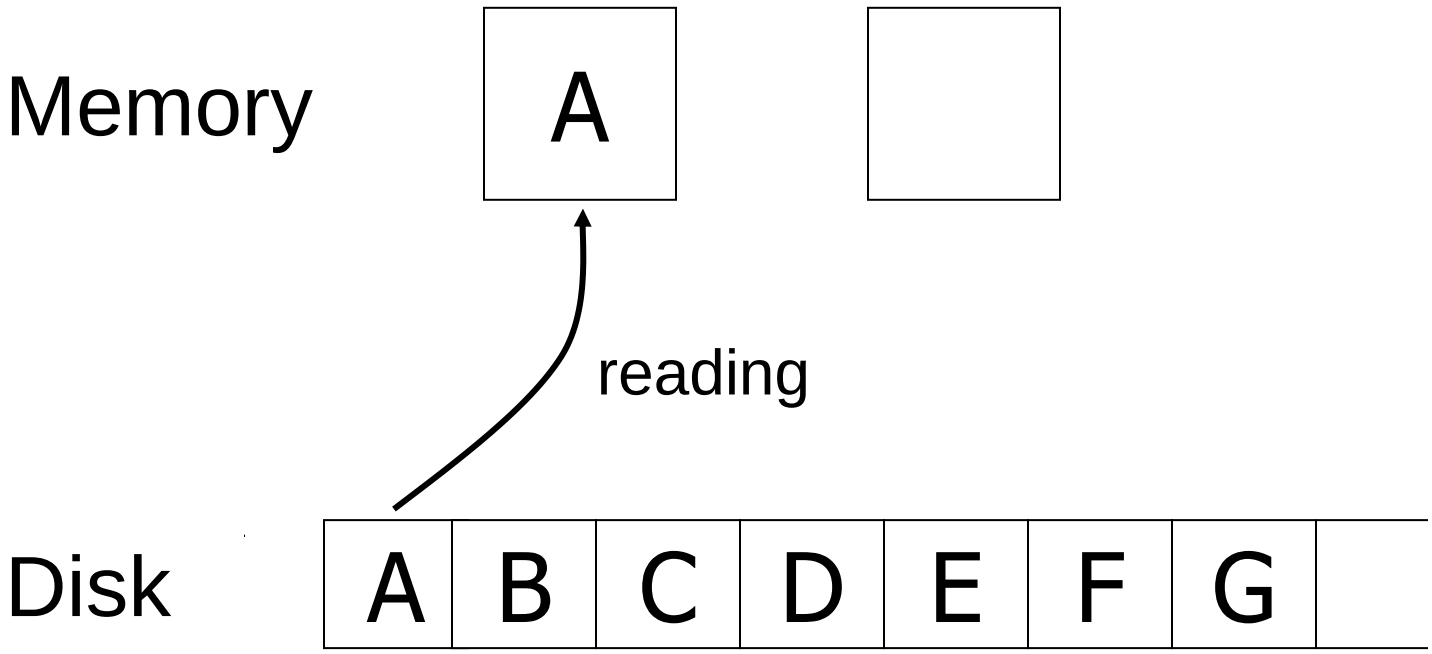
- Task
  - Read block B1 $\rightarrow$ buffer
  - Process data in buffer
  - Read block B2 $\rightarrow$ buffer
  - Process data in buffer
  - …
- Costs
  - P = processing time of a block
  - R = time to read a block
  - n = number of blocks to process
- Single buffer time = n(R+P)

Double Buffering

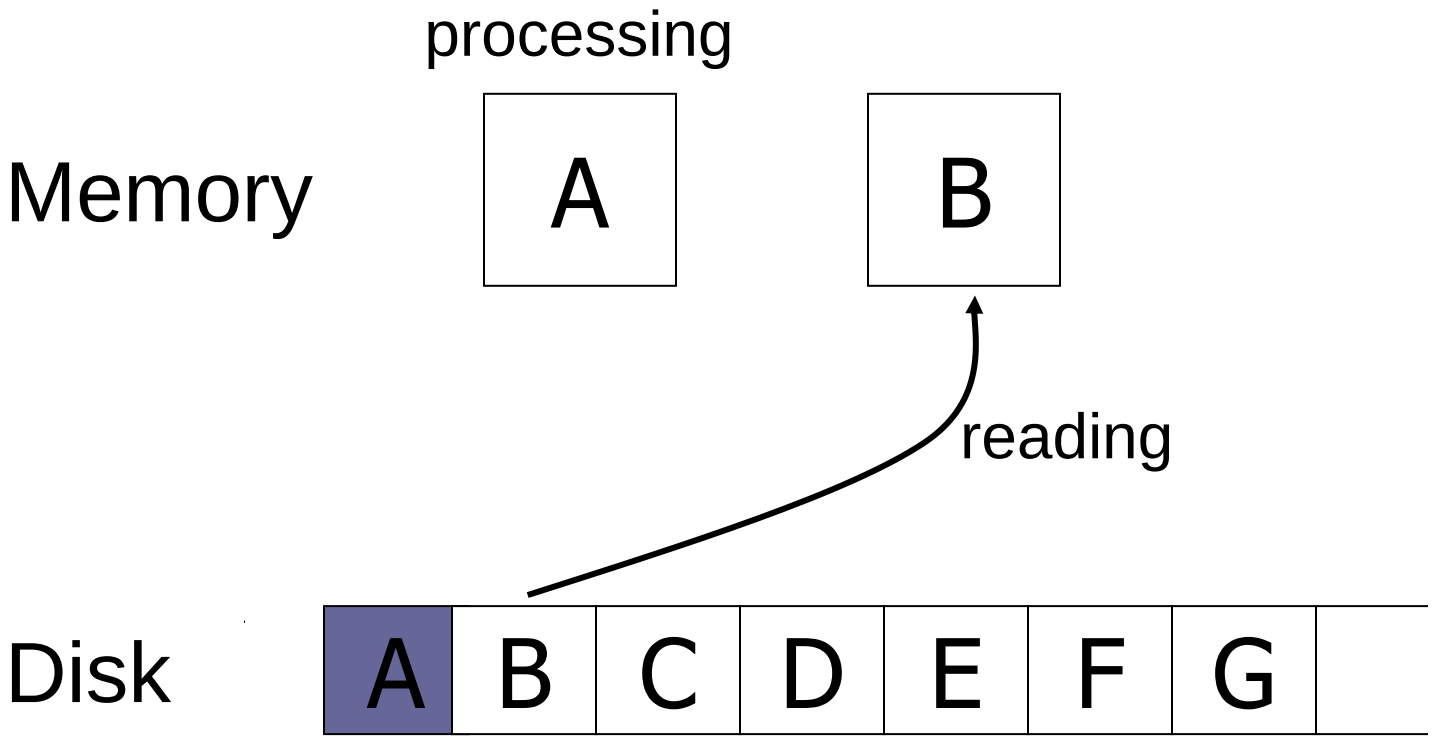
- Two buffers in memory; used alternately



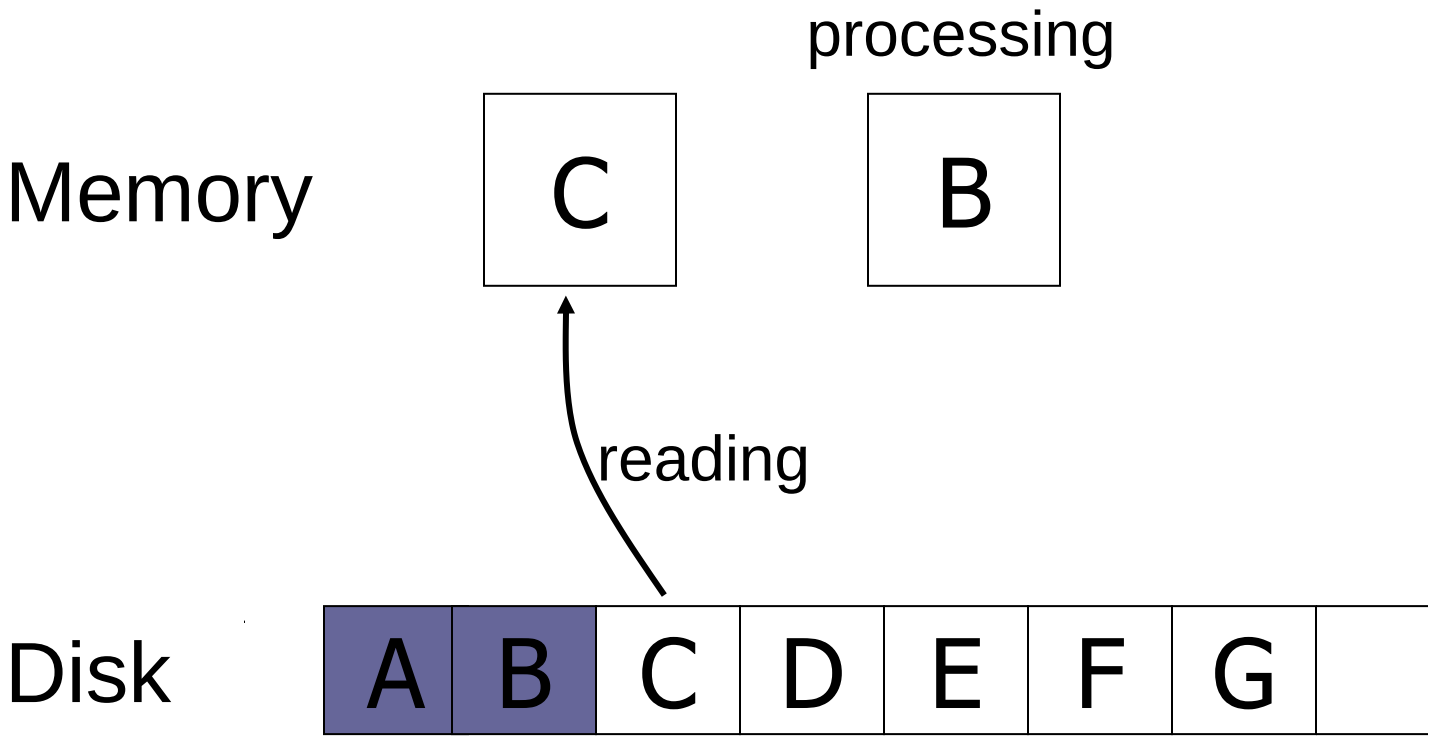
Double Buffering



Double Buffering



Double Buffering



Double Buffering

- Costs
  - P = processing time of a block
  - R = time to read a block
  - n = number of blocks to process
- Single buffer time = n(R+P)
- Double buffer time = R + nP
  - Assuming P ≥ R
  - Otherwise
    - = nR + P

Optimize Disk I/Os

- Access Techniques
  - Minimize random accesses
- *Data volume*
  - *Block size*
- Storage Organization
  - Disk array

Block Size

- Big block $\rightarrow$ amortize I/O costs

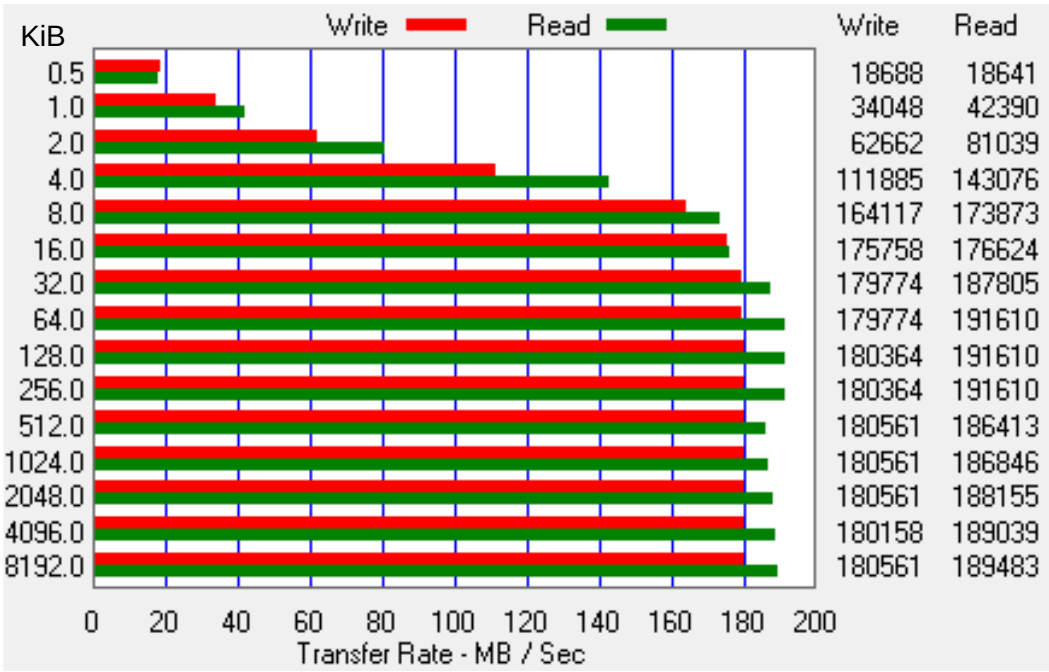
  BUT

- Big block $\rightarrow$ read in more *useless* data; takes longer to read

- Trend
  - As memory prices drop, **blocks get bigger**

Block Size

- ATTO Disk Benchmark
  - 256MB read sequentially block by block
  - No caching
  - Queue length 4



Western Digital 10EZEX 1TB, SATA3, 7200 RPM, sustained transfer rate 150 MB/s

I/Os per second

- IOPS dle HD Tune Pro 5.50
  - Reading 4KiB blocks

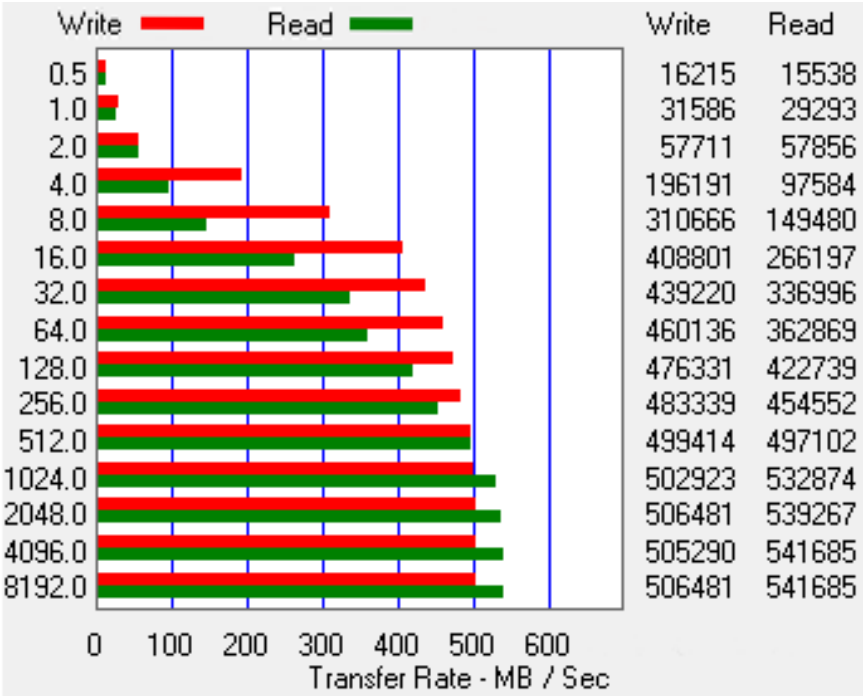
Test	I/O	Time	Transfer
Random seek	65 IOPS	15.269 ms	0.032 MB/s
Random seek 4 KB	65 IOPS	15.427 ms	0.253 MB/s
Butterfly seek	57 IOPS	17.645 ms	0.028 MB/s
Random seek / size 64 KB	64 IOPS	15.522 ms	0.991 MB/s
Random seek / size 8 MB	21 IOPS	47.036 ms	86.209 MB/s
Sequential outer	2798 IOPS	0.357 ms	174.870 MB/s
Sequential middle	2295 IOPS	0.436 ms	143.454 MB/s
Sequential inner	1351 IOPS	0.740 ms	84.414 MB/s
Burst rate	5106 IOPS	0.196 ms	319.134 MB/s

Western Digital 10EZEX 1TB, SATA3, 7200 RPM, sustained transfer rate 150 MB/s

Blocks & IOPS

- Same tests for SSD

Kingston V300 120GB



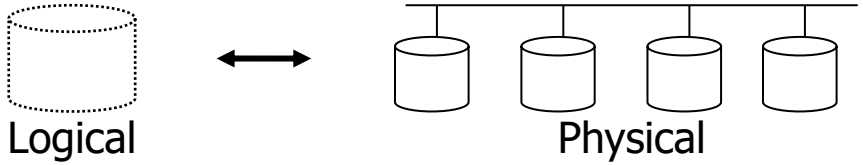
Test	I/O	Time	Transfer
Random seek	5398 IOPS	0.185 ms	2.636 MB/s
Random seek 4 KB	5316 IOPS	0.188 ms	20.764 MB/s
Butterfly seek	5149 IOPS	0.194 ms	2.514 MB/s
Random seek / size 64 KB	4292 IOPS	0.233 ms	65.999 MB/s
Random seek / size 8 MB	118 IOPS	8.461 ms	479.246 MB/s
Sequential outer	3894 IOPS	0.257 ms	243.389 MB/s
Sequential middle	5747 IOPS	0.174 ms	359.183 MB/s
Sequential inner	5816 IOPS	0.172 ms	363.506 MB/s
Burst rate	4194 IOPS	0.238 ms	262.126 MB/s

Optimize Disk I/Os

- Access Techniques
  - Minimize random accesses
- Data volume
  - Block size
- *Storage Organization*
  - *Disk array*

Disk Array

- Multiple disks arranged in one logical drive



  - Increased capacity
  - Parallelized read / write
  - No change in block seek time typically
- Techniques
  - block striping
  - mirroring

Data Striping

- Aims
  - Increase transfer rate by splitting data into multiple disks
  - Parallelize long reads to reduce response time
  - Load balancing $\rightarrow$ increase throughput
  - Decreasing reliability

Data Striping

- Bit-level striping
  - Distribute bits of each byte to among disks
  - Access time worse than that of one disk
  - Rarely used
- Block-level striping
  - *n* disks
  - A block *i* is stored on disk *(i mod n)+1*
  - Reading of different blocks is parallelized
    - If on different disks
  - Large reading may utilize all disks

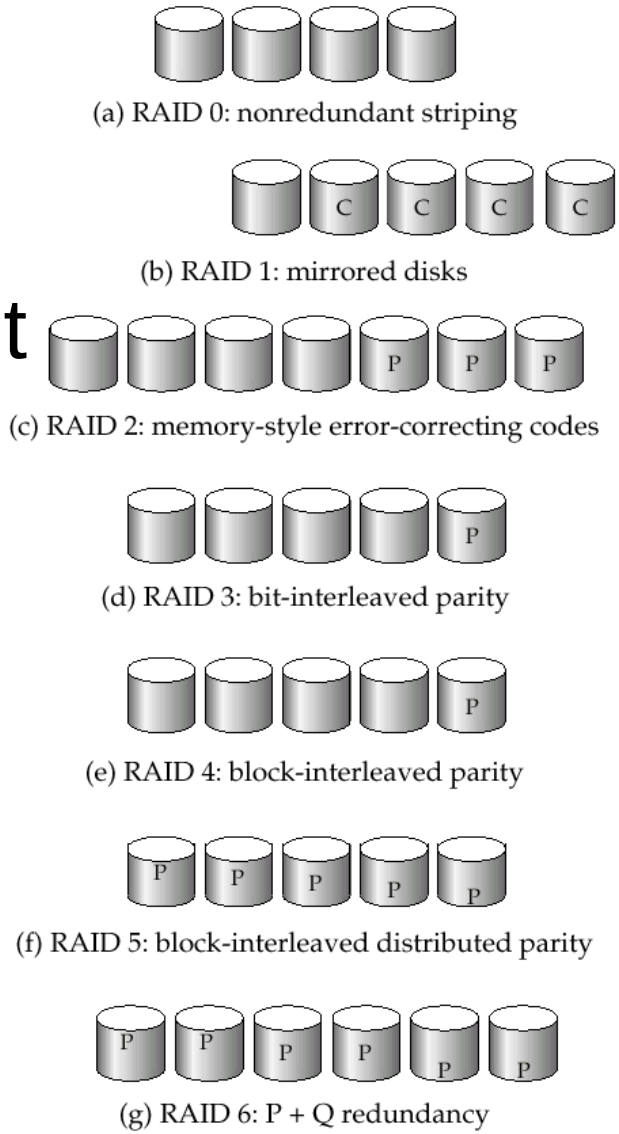
Mirroring

- Increases reliability through replication
  - Logical disk formed by two disks
  - Writing performed to both
  - Reading can be done from any
- Data available of a disk failure
  - Data loss when both disks fail → unlikely
- Beware of dependent failures
  - Fire, electric shock, damage of hardware array controller, …

Skipping details…

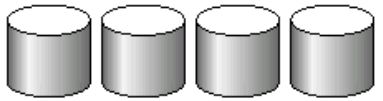
RAID

- Redundant Arrays of Independent Disks
- Different variants for different requirements
  - Different performance
  - Different availability
- Combinations
  - RAID1+0 (or RAID10)
    - RAID0 built over RAID1 arrays



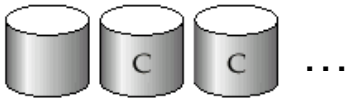
RAID0, RAID1

- RAID0
  - Block striping, non-redundant
  - High performance, non-increased data availability
  - No reduced capacity



(a) RAID 0: nonredundant striping

- RAID1
  - Mirrored disks
    - Sometimes limited to two disks

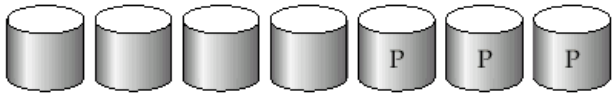


(b) RAID 1: mirrored disks

  - Capacity 1/n; fast reading; writing as of 1 disk
  - Suitable for DB logs, etc.
    - when writes are sequential
  - RAID1E – combines mirroring and striping

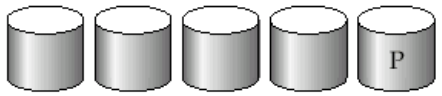
RAID2, RAID3

- RAID2
  - Bit-striping, Hamming Error-Correcting-Code
  - Recovers from of 1 disk failure
  - Error is not detected by the drive!



(c) RAID 2: memory-style error-correcting codes

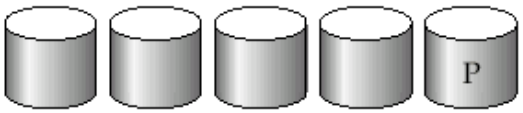
- RAID3
  - Byte-striping with parity
  - 1 parity disk, errors detected by the drive!
  - Writing: calculate and store parity
  - Restoring disk data
    - XOR of bits of the other disks



(d) RAID 3: bit-interleaved parity

RAID4

- Uses block-striping (compared to RAID3)
  - Parity blocks on a separate disk
  - Writing: calculate and store parity
  - Restoring disk data
    - XOR of bits of the other disks
  - Faster than RAID3
    - Block read from 1 disk only → can parallelize
    - Disks may not be fully synchronized



(e) RAID 4: block-interleaved parity

RAID4 (cont.)

- Block write → calculation of parity block
  - Take the original parity, the original block and the new block (2 reads and 2 writes)
  - Or calculate the new parity of all blocks (n-1 reads and 2 writes)
  - Efficient for large sequential writes
- Parity disk is a bottleneck!
  - Writing a block induces writing the parity block
- RAID3, RAID4 – at least 3 disks (2+1)
  - Capacity decreased by the parity disk

RAID5

- Block-Interleaved Distributed Parity
  - Splits data and also parity among *n* disks
  - Load on parity disk of RAID4 removed

(f) RAID 5: block-interleaved distributed parity

  - Parity block for *i*-th block is on disk $\lfloor i/_{n-1} \rfloor \bmod n$
- Example with 5 disks
  - Parity for block *i* is on $\lfloor i/_{4} \rfloor \bmod 5$

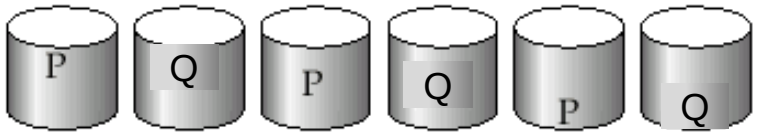
P0	0	1	2	3
4	P1	5	6	7
8	9	P2	10	11
12	13	14	P3	15
16	17	18	19	P4

RAID5 (cont.)

- Faster than RAID4
  - Writing parallel if to different disks
  - Replaces RAID4
    - Same advantages, but removes disadvantage of separate parity disk
- Frequently used solution

RAID6

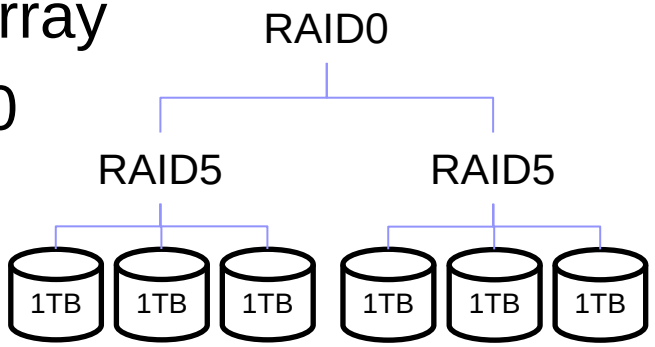
- P+Q Redundancy scheme
  - Similar to RAID5, but stores extra information to recover from failures of more disks
  - Two parity disks (dual distributed parity)
    - Min. 4 disks in array (capacity shrunk by 2 disks)
  - Error-correcting codes (Hamming codes)
    - Repairs failure of two disks



(g) RAID 6: P + Q redundancy

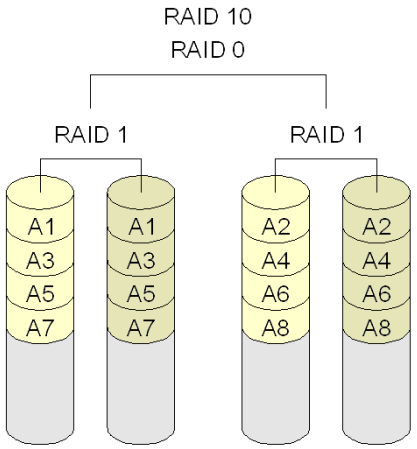
RAID Combinations

- Different variants combined into one system
  - An array assembled from physical disks
  - A resulting array built over these arrays
- Used to increase performance and reliability
- Example
  - RAID5+0 over 6 physical disks
    - Each 3 disks create a RAID5 array
    - RAID5 arrays forms one RAID0

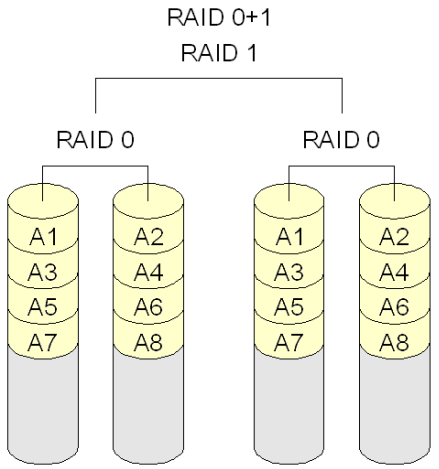


RAID1+0 vs. RAID0+1

- RAID1+0
  - more resistant to failures
  - failure of a disk in any RAID1: OK



- RAID0+1
  - Failure of a disk in 1st RAID0 and failure of a disk in 2nd RAID0 $\Rightarrow$ data loss



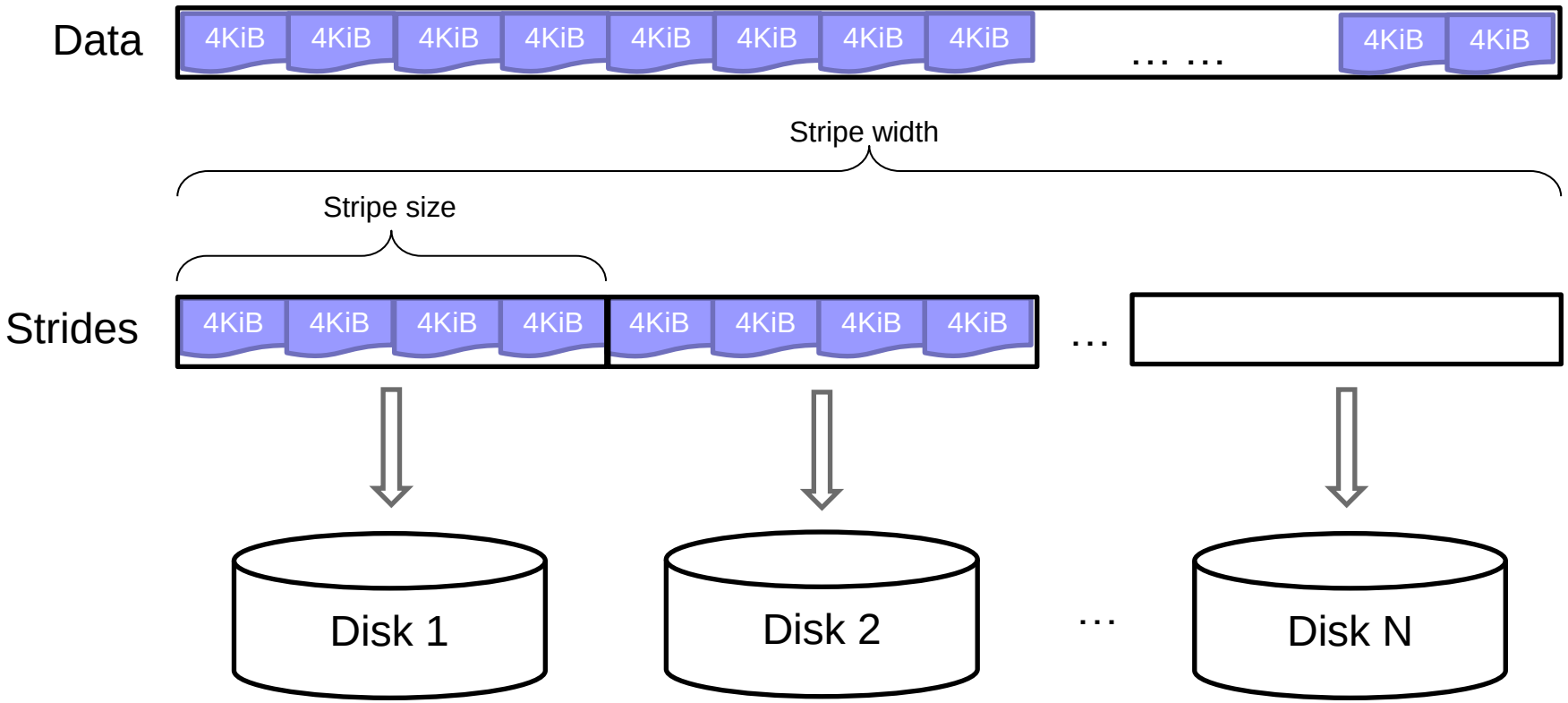
Zdroj: Wikipedia

RAID Organization Parameters

- Data is split into chunks (strides or stripe units)
  - A virtual data unit of data partitioning
- Amount of data in a stripe
  - Number of disks * chunk size
- Example: 4 disks in RAID5
  - Block size = 4 KiB, Stripe size = 512 KiB
    - Stride = 128 (512/4)
  - Stripe-width = 3 * 128

RAID Organization Schema

Example of RAID0 over N disks



RAID Summary

- RAID0 – data availability not important
  - Data can be easily and quickly restored
    - from backup,…
- RAID2, 3 and 4 superseded by RAID5
  - bit-/byte-striping leads to utilization of all disks for any read/write access; non-distributed parity
- RAID6
  - RAID1 and 5 provide sufficient reliability
  - better than RAID5 for high-capacity drives
- Popular combinations:
  - RAID1+0, RAID5+0
- Choice: RAID1+0 vs RAID5

RAID Summary (cont.)

- RAID1+0
  - Much faster writing than RAID5
  - For applications with a large number of writes
  - More expensive than RAID5 (lower capacity)
- RAID5
  - Each writing requires 2 reads and 2 writes typically
    - RAID1+0 needs just 2 writes
  - Suitable for apps with fewer number of writes
  - Check the “chunk” size
- Requirements of current apps in I/Os
  - Very high (e.g., web servers, …)
  - Need to buy many disks to fulfill the requirements
    - If their capacity is sufficient, use RAID1 (no further costs)
    - Preferably RAID1+0

RAID Summary (cont.)

- Does not substitute backups!!!
- Implementation
  - SW – supported in almost any OS
  - HW – special disk controller
    - Necessary to use battery-backup cache or non-volatile RAM
    - Double-check controller's CPU performance and tune stripe size.
- Hot-swapping
  - Usually supported in HW implementations
  - No problem in SW implementation, if disk supports
- Spare disks
  - Presence of extra disks in array

Disk Failures

- Intermittent failure
  - Error during read/write → repeat → OK
- Medium defect
  - Permanent fault of a sector
  - Modern disks detect and correct
    - Allocated from a spare capacity
- Permanent failure
  - Total damage → replace the disk

Coping with Disk Failures

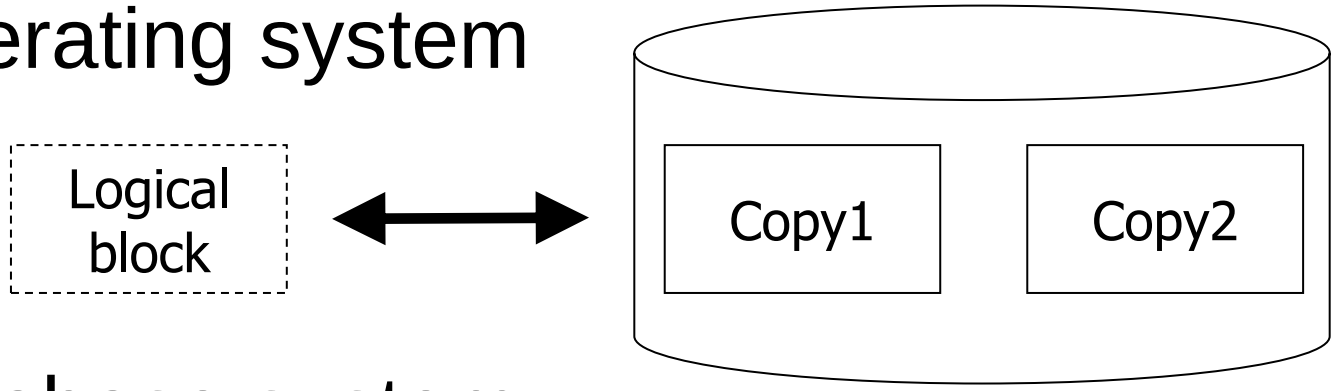
- Detection
  - Checksum
- Correction by redundancy
  - Stable storage
    - Disk array
    - Store at more places of the same disk
      - super-block; ZFS on data
    - Journal (log / journal of modifications)
  - Error correcting codes (ECC)
    - Hamming codes, …

Journaling in File Systems

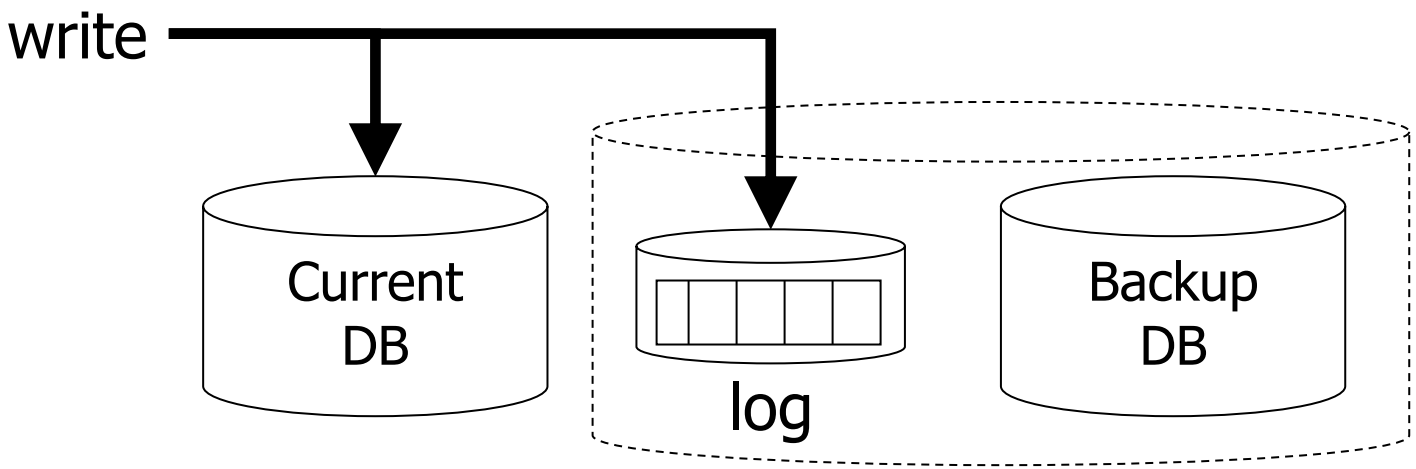
- Storing a data block:
  1. Add an unused block to the list of used space
  2. Write data block
  3. Write file metadata referencing that data block
- Modern FS uses journaling
  - Start transaction in journal
  - Store info about steps 1.-3. to journal
  - Do steps 1.-3.
  - End transaction in journal

Stable Storage and Databases

- Operating system



- Database system



Skipping ECC…

File System Tuning for DBMS

- FS block size ≤ DB block size
  - ZFS has 128KB by default!
- DB journal (WAL in PostgreSQL)
  - ext2; ext3/4 with data=writeback (journal off)
- DB data
  - ext3/4 with data=ordered (only metadata journaled)
- Switch off *file access times* (noatime)
- Eliminate swapping (vm.swappiness = 0)
- Process memory allocation (vm.overcommit_memory = 2)
- …

Error Correcting Codes

- Parity bit = even / odd parity
  - Used in RAID3,4,5
- Example of even parity
  - RAID4 over 4 disks, block no. 1:

```
Disk 1: 11110000…        Disk 1: 11110000…
Disk 2: 10101010…        Disk 2: ????????…
Disk 3: 00111000…        Disk 3: 00111000…
Disk P: 01100010…        Disk P: 01100010…
```

Error Correcting Codes

- Algebra with operator sum modulo-2
  - Even parity, i.e., adding 1 to make even number of 1's
  - $\vec{x} \oplus \vec{y} = \vec{y} \oplus \vec{x}$
  - $\vec{x} \oplus (\vec{y} \oplus \vec{z}) = (\vec{x} \oplus \vec{y}) \oplus \vec{z}$
  - $\vec{x} \oplus \vec{0} = \vec{x}$
  - $\vec{x} \oplus \vec{x} = \vec{0}$
- If $\vec{x} \oplus \vec{y} = \vec{z}$, then $\vec{y} = \vec{x} \oplus \vec{z}$
  - Add $\vec{x}$ to both sides…

Error Correcting Codes

- Hamming code
  - Example to recover from 2 crashes
    - 7 disks -> four data disks
  - Redundancy schema:
    - Parity disk contains even parity
    - Parity computed from data disks denoted by 1

	Data				Parity		
Disk No:	1	2	3	4	5	6	7
	1	1	1	0	1	0	0
	1	1	0	1	0	1	0
	1	0	1	1	0	0	1

Error Correcting Codes (cont.)

- Hamming code
  - Content sample and writing

Disk No:	Data 1	2	3	4	Parity 5	6	7
	1	1	1	0	1	0	0
	1	1	0	1	0	1	0
	1	0	1	1	0	0	1

```
Disk 1: 11110000…
Disk 2: 10101010…
Disk 3: 00111000…
Disk 4: 01000001…
Disk 5: 01100010…
Disk 6: 00011011…
Disk 7: 10001001…
```

```
Disk 1: 11110000…
Disk 2: 00001111…
Disk 3: 00111000…
Disk 4: 01000001…
Disk 5: 11000111…
Disk 6: 10111110…
Disk 7: 10001001…
```

Writing to disk 2:
00001111…

Error Correcting Codes (cont.)

- Hamming code
  - Disk failure

Disk No:	Data 1	2	3	4	Parity 5	6	7
	1	1	1	0	1	0	0
	1	1	0	1	0	1	0
	1	0	1	1	0	0	1

```
Disk 1: 11110000…        Disk 1: 11110000…
~~Disk 2:~~ ????????…    Disk 2: 00001111…
Disk 3: 00111000…        Disk 3: 00111000…
Disk 4: 01000001…        Disk 4: 01000001…
~~Disk 5:~~ ????????…    Disk 5: 11000111…
Disk 6: 10111110…        Disk 6: 10111110…
Disk 7: 10001001…        Disk 7: 10001001…
```

Recovery of disk 2
(row in redund. schema with 0 for disk 5)

Recovery of disk 5

Error Correcting Codes (cont.)

- Definition of Hamming Code
  - A *code* of length *n* is a set of *n*-bit vectors (*code words*).
  - *Hamming distance* is the count of different values in two n-bit vectors.
  - *Minimum distance of a code* is the smallest Hamming distance between any different code words.
  - *Hamming code is a code with min. dist. "3"*
    - Up to two bit-flips can be detected (but not corrected).
    - 1 bit-flip is detected and corrected.

Error Correcting Codes (cont.)

- Generating Hamming Code (n,d); p=n-d
  - Number bits from 1, write them in cols in binary
  - Every column with one ‘X’ (single bit set) is parity
  - Row shows the sources for parity computation
  - Column shows which parity bits cover data bit.

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Data/parity bits		p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7	d8	d9	d10	d11	p5	d12
Parity bit coverage	p1 2^0	X		X		X		X		X		X		X		X		X
	p2 2^1		X	X			X	X			X	X			X	X		
	p3 2^2				X	X	X	X					X	X	X	X		
	p4 2^3								X	X	X	X	X	X	X	X		
	p5 2^4																X	X

Error Correcting Codes (cont.)

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Data/parity bits		p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7	d8	d9	d10	d11	p5	d12
Parity bit coverage	p1 2^0	X		X		X		X		X		X		X		X		X
	p2 2^1		X	X			X	X			X	X			X	X		
	p3 2^2				X	X	X	X					X	X	X	X		
	p4 2^3								X	X	X	X	X	X	X	X		
	p5 2^4																X	X

- Store data bits 1010 in Hamming Code (7,4)
- To correct errors in data read from storage:
  - Check all parity bits.
  - Sum the positions of bad ones to get address of the wrong bit.
- Examples:
  - 1111010 → one bit was flipped, so it can be corrected.
  - 1011110
  - 1001110 → two bits were flipped, so it cannot distinguish 2-bit and 1-bit error.
  - 1110000 → three bits were flipped here

Error Correcting Codes (cont.)

- Extended Hamming Code
  - Add an extra parity bit over all bits
    - To tell even or odd number of error

Bit position		0	1	2	3	4	5	6	7
Data/parity bits		pX	p1	p2	d1	p3	d2	d3	d4
Parity bit coverage	p1 2^0		X		X		X		X
	p2 2^1			X	X			X	X
	p3 2^2					X	X	X	X
	pX 0	X	X	X	X	X	X	X	X

- Store data bits 1010 in Extended Hamming Code (7,4)
- Detect/correct error if any:
  - 01111010
  - 01011110
  - 01001110 → 2 bits were flipped; but no clue which.
  - 01110000 → odd number of bits **(>1)** were flipped; but no clue which.

Error Correcting Codes (cont.)

- Reed-Solomon Code (n,d)
  - ECC adding *t* check bits to data bits
  - Can detect up to *t* bit errors
  - Can correct up to $\lfloor t/2 \rfloor$ errors
  - So, min. Hamming distance is $\mathrm{t} = n - d + 1$.

Failures – Terminology

- Mean Time To Failure (MTTF)
  - Also: Mean Time Between Failures (MTBF)
  - Corresponds to failure likelihood
  - Average operating time between failures
    - Half of disks fails during the period
    - Assumes uniform distribution of failures
  - Decreases with disk age
  - Usually, 1 000 000 hours and more
    - $\Rightarrow$ 114 years
    - i.e., it fails with 100% in 228 years

      $\Rightarrow P_{\text{failure in a year}}=0,44\%$
    - $\Rightarrow$ **Annualized Failure Rate (AFR)**

Failures – Terminology (cont.)

- Example
  - MTTF = 1 000 000 hours
  - $\Rightarrow$ population of 2 000 000 disks
    - One fails per hour, i.e., 8,760 disks a year
    - $\Rightarrow$ probability of failure in a year = 0,44%

Failures – Terminology (cont.)

- Alternative measure
  - Annualized Failure Rate **(AFR)**
  - Component Design Life

- Annual Replacement Rate (ARR)

  or Annualized Return Rate
  - Not all failures caused by disk faults
    - Defective cable, etc.
  - It is stated
    - 40% of ARR is “No Trouble Found” (NTF)
    - AFR = ARR*0.6 ARR = AFR / 0.6

Skipping estimates…

Failures and Manufacturers

- Seagate http://www.seagate.com/docs/pdf/whitepaper/drive_reliability.pdf (November 2000)
  - Savvio® 15K.2 Hard disks – 73 GB
    - AFR = 0,55%
  - Seagate estimates MTTF for a disk as the number of power-on hours (POH) per year divided by the first-year AFR.
  - AFR is derived from Reliability-Demonstration Tests (RDT)
    - RDT at Seagate = hundreds of disks operating at 42ºC ambient temperature

Failures and Manufacturers

- Influence of temperature to MTTF during 1st year
  - Seagate

Temp (°C)	Acceleration Factor	Derating Factor	Adjusted MTTF
25	1.0000	1.00	232,140
26	1.0507	0.95	220,533
30	1.2763	0.78	181,069
34	1.5425	0.65	150,891
38	1.8552	0.54	125,356
42	2.2208	0.45	104,463
46	2.6465	0.38	88,123
50	3.1401	0.32	74,284
54	3.7103	0.27	62,678
58	4.3664	0.23	53,392
62	5.1186	0.20	46,428
66	5.9779	0.17	39,464
70	6.9562	0.14	32,500

Failures and Manufacturers

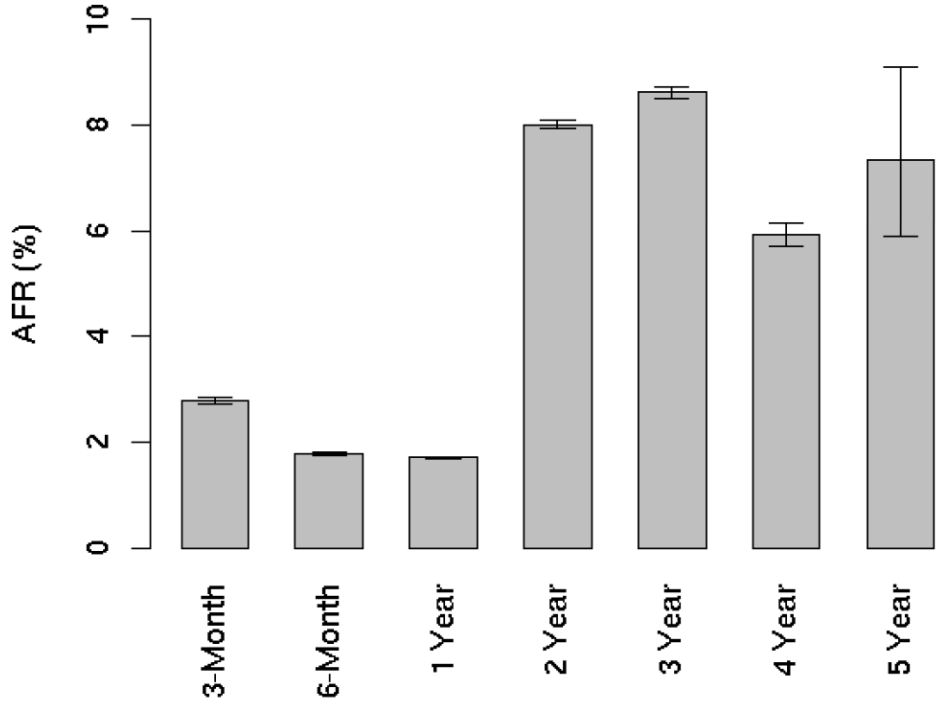
- Seagate Barracuda ES.2 Near-Line Serial ATA disk

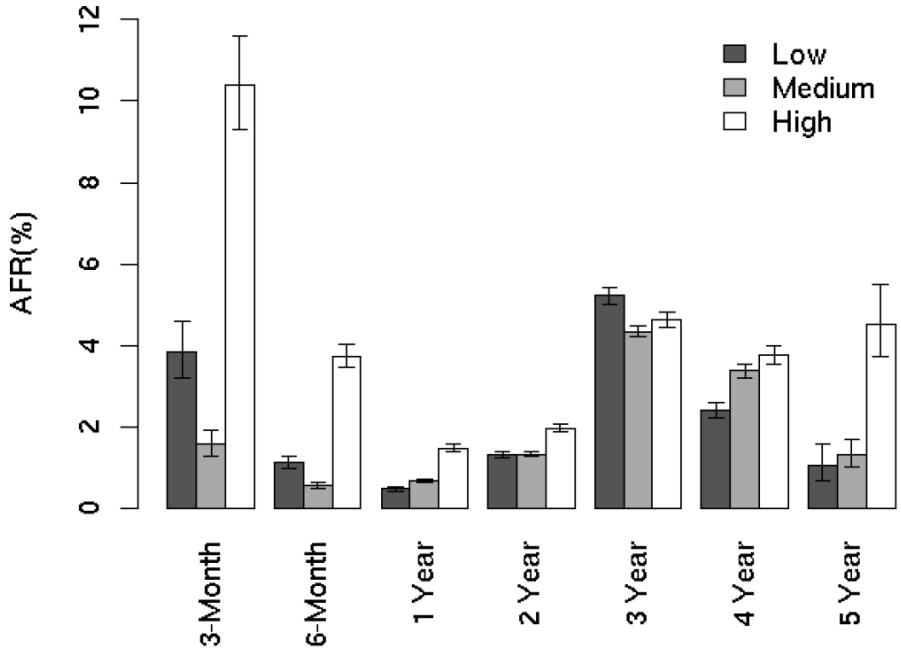
		MODEL:					
		Weibull		Warranty Data (OEM only)		Flatline Model	
Year	Cumulative power-on hours	Yearly failure rate	Cumulative failure rate	Yearly failure rate	Cumulative failure rate	Yearly failure rate	Cumulative failure rate
1	2,400	1.20%	1.20%	1.20%	1.20%	1.20%	1.20%
2	4,800	0.55%	1.75%	0.78%	1.98%	0.55%	1.75%
3	7,200	0.43%	2.18%	0.39%	2.37%	0.55%	2.30%
4	9,600	0.37%	2.55%			0.55%	2.86%
5	12,000	0.33%	2.88%			0.55%	3.41%
6	14,400	0.30%	3.18%			0.55%	3.96%
7	16,800	0.28%	3.46%			0.55%	4.51%
8	19,200	0.26%	3.72%			0.55%	5.06%
9	21,600	0.24%	3.96%			0.55%	5.62%
10	24,000	0.23%	4.19%			0.55%	6.17%

Note1: Weibull – stat. method for modeling progress of failures
Note2: 2400 hours/yr => 6.5 hrs a day!

Failures – practice

- Google http://research.google.com/archive/disk_failures.pdf (FAST conference, 2007)
  - Test on 100,000 disks
    - SATA, PATA disks; 5400-7200 rpm; 80-400 GB

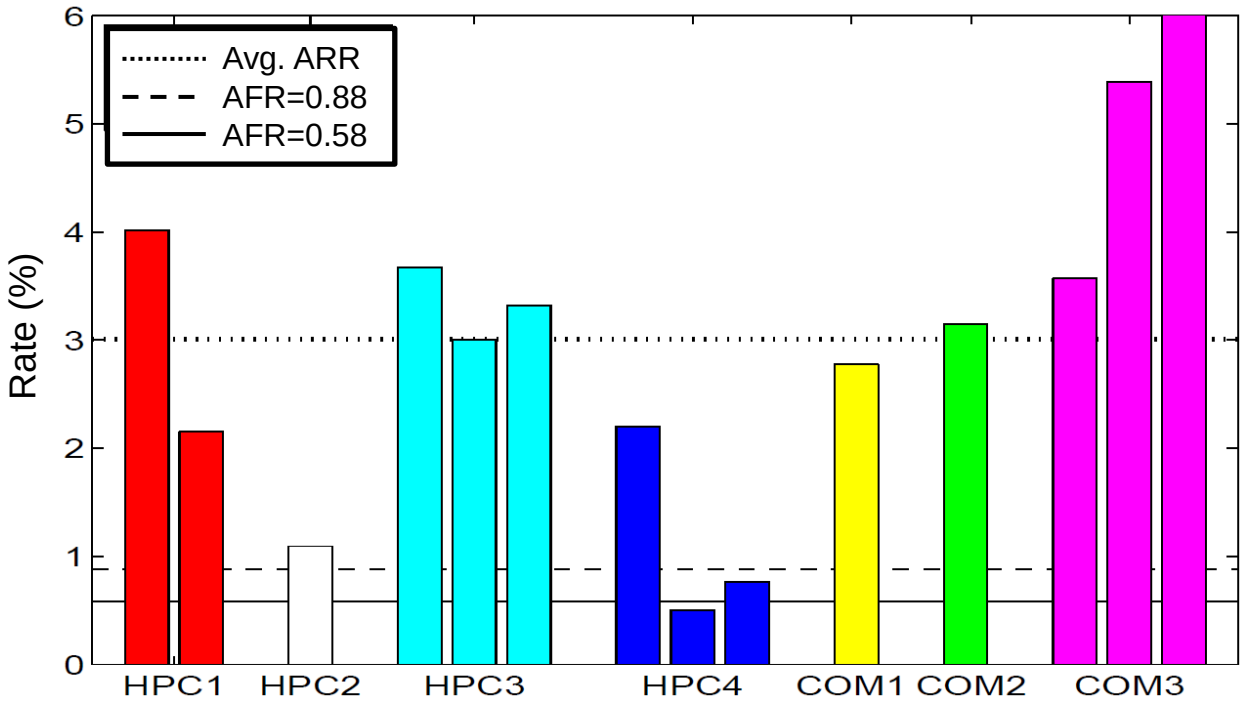




Failures – practice

- Study on 100,000 SCSI, FC, SATA disks

  http://www.cs.cmu.edu/~bianca/fast07.pdf (FAST conference, 2007)



HPC3: 3064x SCSI disk, 15k rpm, 146GB
11000x SATA disk, 7200 rpm, 250GB
144x SCSI disk, 15k rpm, 73GB

Failures – practice

- Conclusion:
  - AFR increases with temperature raising
    - Not confirmed in data by Google
  - SMART parameters are well-correlated with higher failure probabilities
    - Google
      - After the *first scan error*, a disk is 39 times more likely to fail within 60 days.
      - First *errors in reallocations*, *offline reallocations*, and *probational counts* are strongly correlated to higher failure probabilities.
  - Appropriate to use AFR 3-4% in evaluations
    - If you plan on AFR that is 50% higher than MTTF suggests, you'll be better prepared.
  - Be ready to replace after 3 yrs of operation

Failure Recovery

- We know AFR = 1 / (2*MTTF)
- Mean Time To Repair (MTTR)
  - Time from failure to recovery of operation
  - = time to replace the failing unit + data recovery
  - $P_{\text{Failure During Repair}} = P_{FDR} = (2*MTTR) / 1 \text{ year}$
    - Assuming: very short time
- Mean Time To Data Loss (MTTDL)
  - Depends on MTTF and MTTR
  - Mean time between two data-loss events
  - For one disk (i.e., data stored in one disk)
    - MTTDL = MTTF

Check units! Years vs. hours

Failure Recovery – Set of disks

- Assumption
  - Failure of each disk is equally probable and independent of each other
- Example for RAID0
  - One disk
    - $AFR_{1\ disk}$ = 0,44% (MTTF = 1,000,000 hrs. = 114 yrs.)
  - System of 114 disks ($MTTF_{100\ disks} = MTTF_{1\ disk} / 100$)
    - $AFR_{100\ disks}$ = 44% (MTTF = 10,000 hrs. = 1.14 yrs.)
      - 1 disk fails each year on *average*
    - Probability (exactly *1* out of *n* vs. at least *1* out of *n* fail)
      - $P_{exactly\ 1\ of\ 100\ fails}$ = 28,43% $\mathbf{P_{at\ least\ 1\ of\ 100\ fails} = 35,66\%}$
      - $P_{exactly\ 1\ of\ 10\ fails}$ = 4,23% $\mathbf{P_{at\ least\ 1\ of\ 10\ fails} = 4,31\%}$
- $AFR_{n\ disks} = AFR_{1\ disk} * n$
- MTTDL = 0.5 / AFR
  - e.g., $MTTDL_{100\ disks}$ = 0.5/0.44 = 1.136 yrs

RAID1: Example of Reliability

- 2 mirrored 500GB disks
  - AFR of each = 3%
- Replacement of failed and array recovery in 3 hrs
  - MTTR = 3 hrs (at 100MB/s copying takes 1.5 hrs.)
- Probability of data loss:
  - $P_{\text{1-disk failure}} = AFR = 0.03$
  - $P_{\text{1-out-of-2 failure}} = 0.06$
  - $P_{FDR} = 2 * MTTR / 1\ year = 2*3 / 8760 = 0{,}000\ 685$
  - $P_{\text{data loss}} = P_{\text{1-out-of-2 failure}} * P_{FDR} * P_{\text{1-disk failure}}$
    $= 0{,}000\ 001\ 233$
  - **MTTDL** $= 0.5 / P_{\text{data loss}} = 405\ 515$ yrs

RAID0: Example of Reliability

- 1 disk AFR = 3% ($P_{1\text{-disk failure}}$)
- RAID0 – two disks, striping
  - $P_{\text{data loss}} = P_{1\text{-out-of-2 failure}} = 6\%$
  - MTTDL = 0.5 / (0.06) = 8.3 yrs
    - i.e., $AFR_{\text{array}} = 6\%$

RAID4: Example of Reliability

- 1 disk AFR = 3% ($P_{1\text{-disk failure}}$)
- RAID4 – repairs failure of 1 disk
  - 4 disks (3+1)
  - MTTR = 3 hrs
    - P_{FDR} = 2*3 / 8760 = 0,000 685
  - $P_{\text{data loss}} = P_{\text{1-out-of-4 failure}} * P_{FDR} * P_{\text{1-out-of-3 failure}}$
  - $P_{\text{data loss}}$ = 4*0,03 * 2/2920 * 3*0,03
    = 0,000 007 397
    - which is AFR of this array
  - MTTDL = 0.5 / $P_{\text{data loss}}$ = 67 592 yrs

RAID6: Example of Reliability

- RAID6 – repairs failure of 2 disks
  - 4 disks (2+2)
  - $P_{\text{data loss}} = P_{\text{1-out-of-4 failure}} * P_{\text{FDR}} * P_{\text{RAID4over3}}$

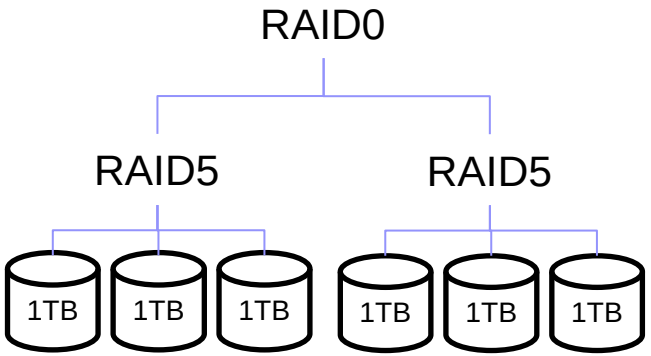
Array Reliability

- n disks
  - disks in array in total (incl. parity disks)
- 1 parity disk
  - ensures data redundancy
  - $AFR_{array1p} = n*AFR_{1\ disk} * P_{FDR} * (n-1)*AFR_{1\ disk}$
  - $MTTDL = 0.5 / AFR_{array}$
- 2 parity disks
  - $AFR_{array2p} = n*AFR_{1\ disk} * P_{FDR} * AFR_{array1p}$

For N-1 disks!

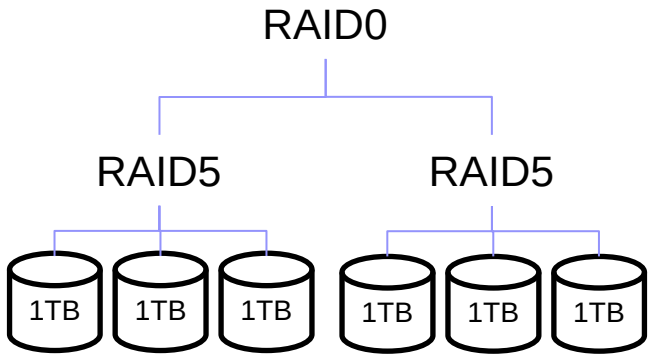
Example of Reliability: RAID Combinations

- Combination of arrays
  - Evaluate MTTDL for individual components
    - Use it as MTTF of a virtual disk
  - Evaluate the final MTTDL

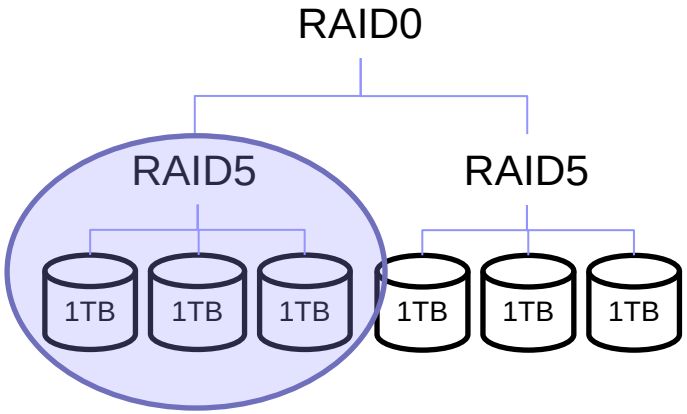


Example of Reliability: RAID Combinations

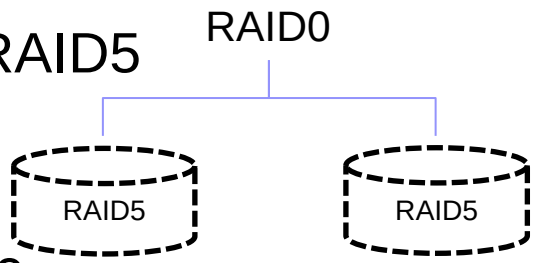
- RAID5+0
  - 1 disk: AFR_{disk}



1) Evaluate AFR_{RAID5}



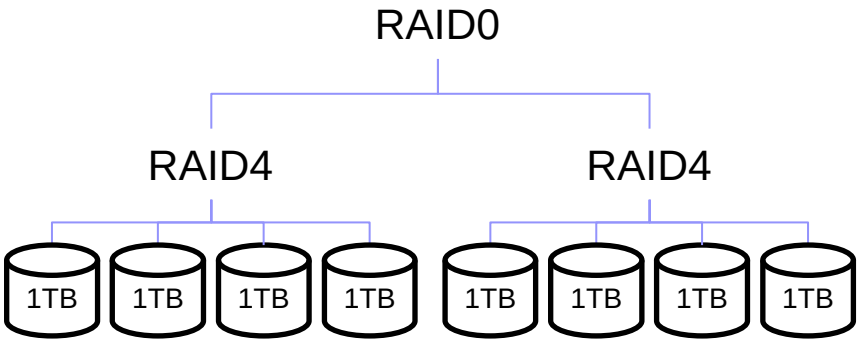
2) Evaluate $AFR_{RAID0} = 2 * AFR_{RAID5}$



3) $MTTDL_{RAID5+0} = 0.5 / AFR_{RAID0}$

Example of Reliability: RAID Combinations

- RAID4+0 over 8 disks
  - 1 disk AFR=3%, MTTR = 3 hrs.



  - Assemble one RAID4 over every 4 disks
    - $AFR_{RAID4} = 4*AFR * P_{FDR} * 3*AFR = \ldots = 7.4*10^{-6}$
  - Assemble two RAID4s into RAID0
    - $AFR_{RAID4+0} = 2 * AFR_{RAID4} = 1.48*10^{-6}$
    - $MTTDL = 0.5 / AFR_{RAID4+0} = 33\ 796$ yrs.

Failures in RAID

- *Data is not written to all disks.*
  - „**Write Hole**“ Phenomenon
  - Severity due to being unnoticed
    - Discoverable during array reconstruction
- Solution
  - UPS
  - Journaling
    - but with “data written” commit message (-:
  - Synchronize the array (data scrubbing)
  - Special file system (ZFS)
    - uses "copy-on-write" to provide write atomicity

RAID over SSD

- SSD – issue of wearing
  - Limited writes is handled by moving writes to other areas, i.e., wear-leveling
  - Consequence: total failure after some time
- RAID over SSD
  - Worse data availability/reliability
    - Almost sure that SSDs fail at once
  - Diff-RAID
    - Distributes parity unevenly
    - After replacing a failed SSD with a brand-new one, parity is moved primarily to the most worn-out drive.

Recommended Reading

- Dual parity
  - https://www.kernel.org/pub/linux/kernel/people/hpa/raid6.pdf
- Software RAID in SUSE
  - https://www.suse.com/documentation/sles10/stor_admin/data/raidevms.html
  - Sections:
    - **Managing Software RAIDs with EVMS**
    - **Managing Software RAIDs 6 and 10 with mdadm**
- SSD on Wikipedia
  - https://en.wikipedia.org/wiki/Solid-state_drive
- Živě.cz: Ze světa: kolik reálně zapsaných dat vydrží moderní SSD? (in Czech)
  - http://m.zive.cz/ze-sveta-kolik-realne-zapsanych-dat-vydrzi-moderni-ssd/a-177557/?textart=1
- Chunk size and performance
  - https://raid.wiki.kernel.org/index.php/Performance

Takeaways

- Understating of IOPS
  - Atomic unit of data access
  - How to increase performance
    - Double buffering, RAID principles
- Failures
  - Terminology: MTTR, MTTF, AFR
  - Issues & solutions
    - RAID: Write-hole phenomenon
    - Implications of using SSDs in arrays
    - Journaling, …