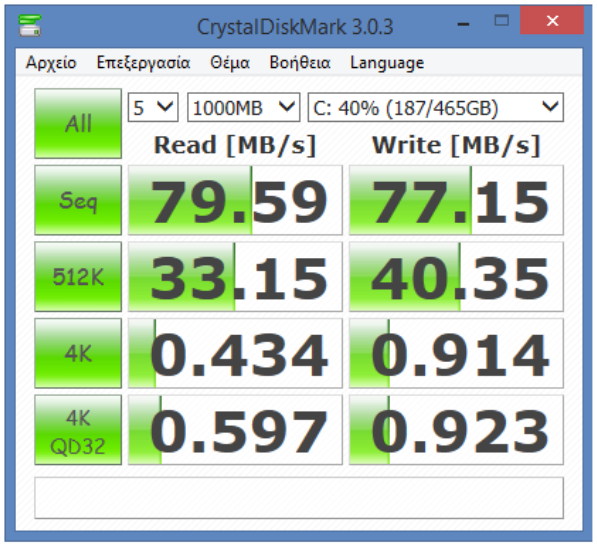
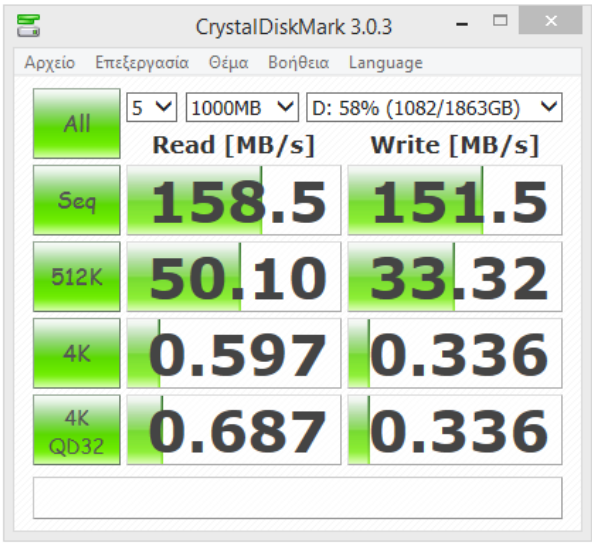
Μέσα Αποθήκευσης

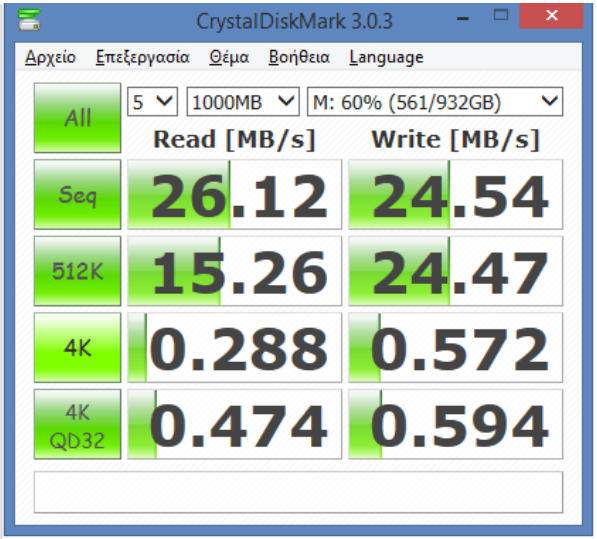
Ιωάννης Κωτίδης



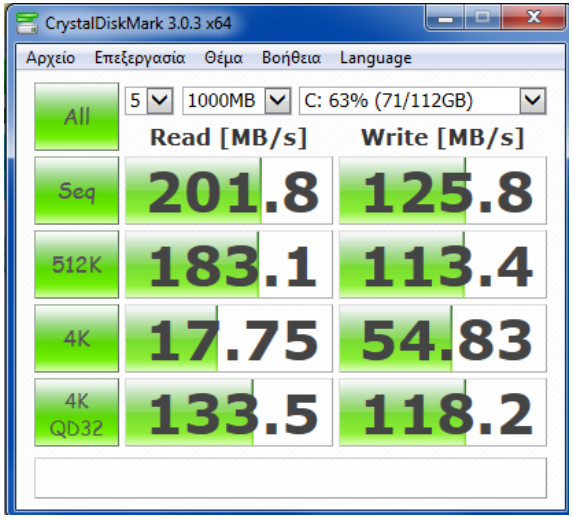
hdd



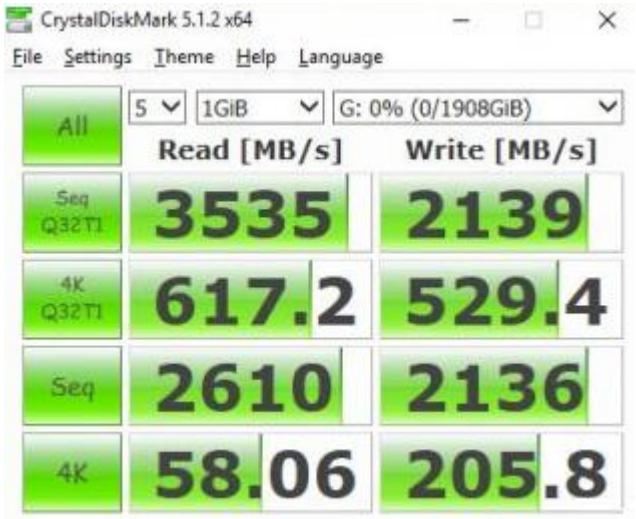
hdd



external hdd



SSD (sata2)

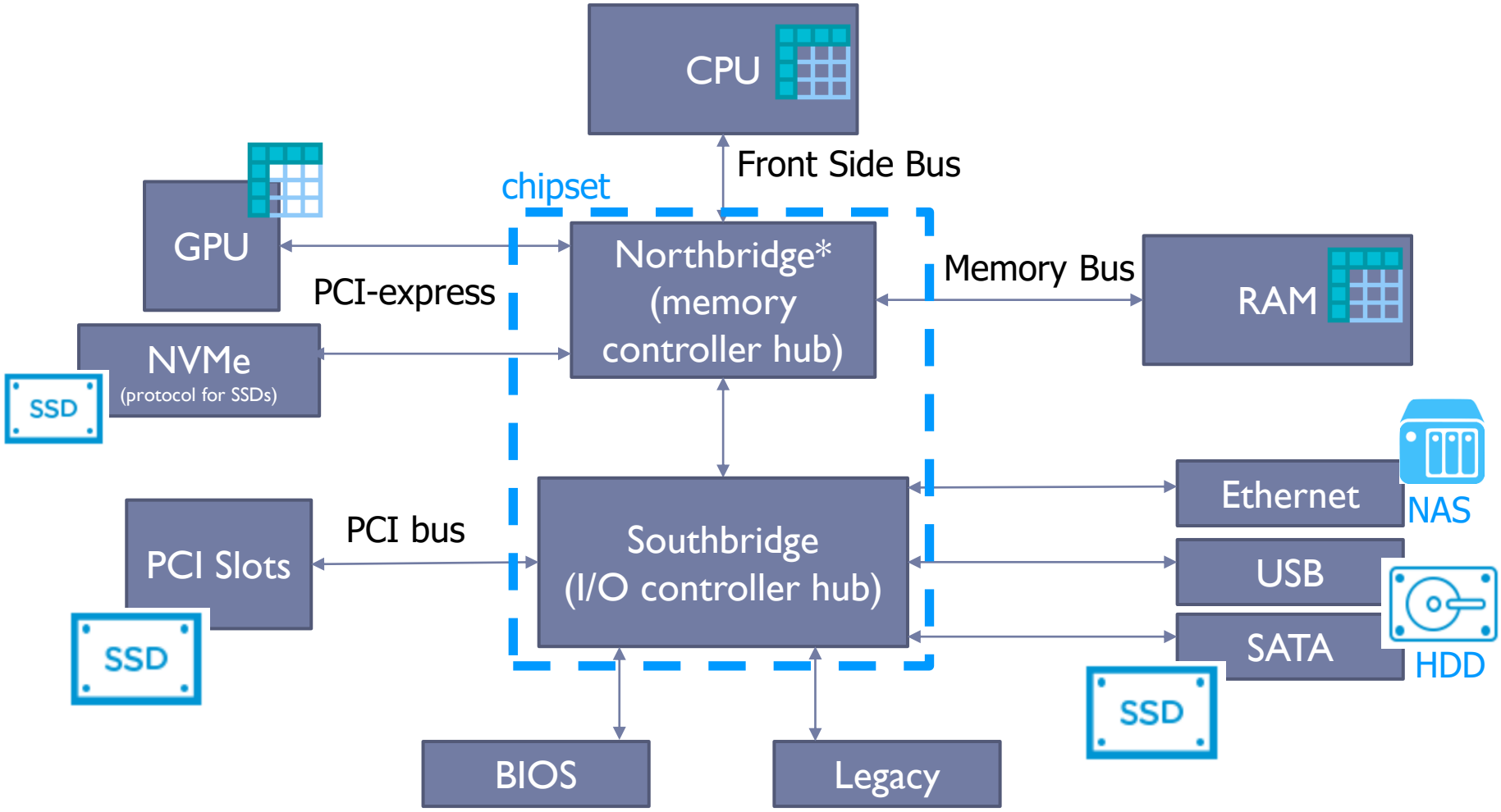


SSD (NVMe)

Περιγραφή

- Αρχιτεκτονική Η/Υ
- Ιεραρχία Μνήμης
- Ο νόμος του Moore
- Δίσκοι, λειτουργία, παραδείγματα
- Αντιμετώπιση σφαλμάτων
- 0 κανόνας των 5 λεπτών
- Βελτιστοποιήσεις
- SSDs

Personal Computer Architecture (generic)



*memory controller is often integrated into modern CPUs

Παράγοντες/χαρακτηριστικά

- Ταχύτητα μέσου
- Κόστος αποθήκευσης (ενδεικτικές τιμές - outdated)
  - Μαγνητικοί δίσκοι: 3-10 λεπτά/GB
  - Solid State Disks (SSD): 16-30 λεπτά/GB
  - RAM: 8-15 ευρώ/GB
- Αξιοπιστία
  - Απώλεια δεδομένων σε περίπτωση διακοπής παροχής τάσης
  - Απώλεια λόγω σφάλματος υλικού
- Κατανάλωση ενέργειας, απαιτήσεις σε ψύξη
- Θόρυβος (laptop/desktop)

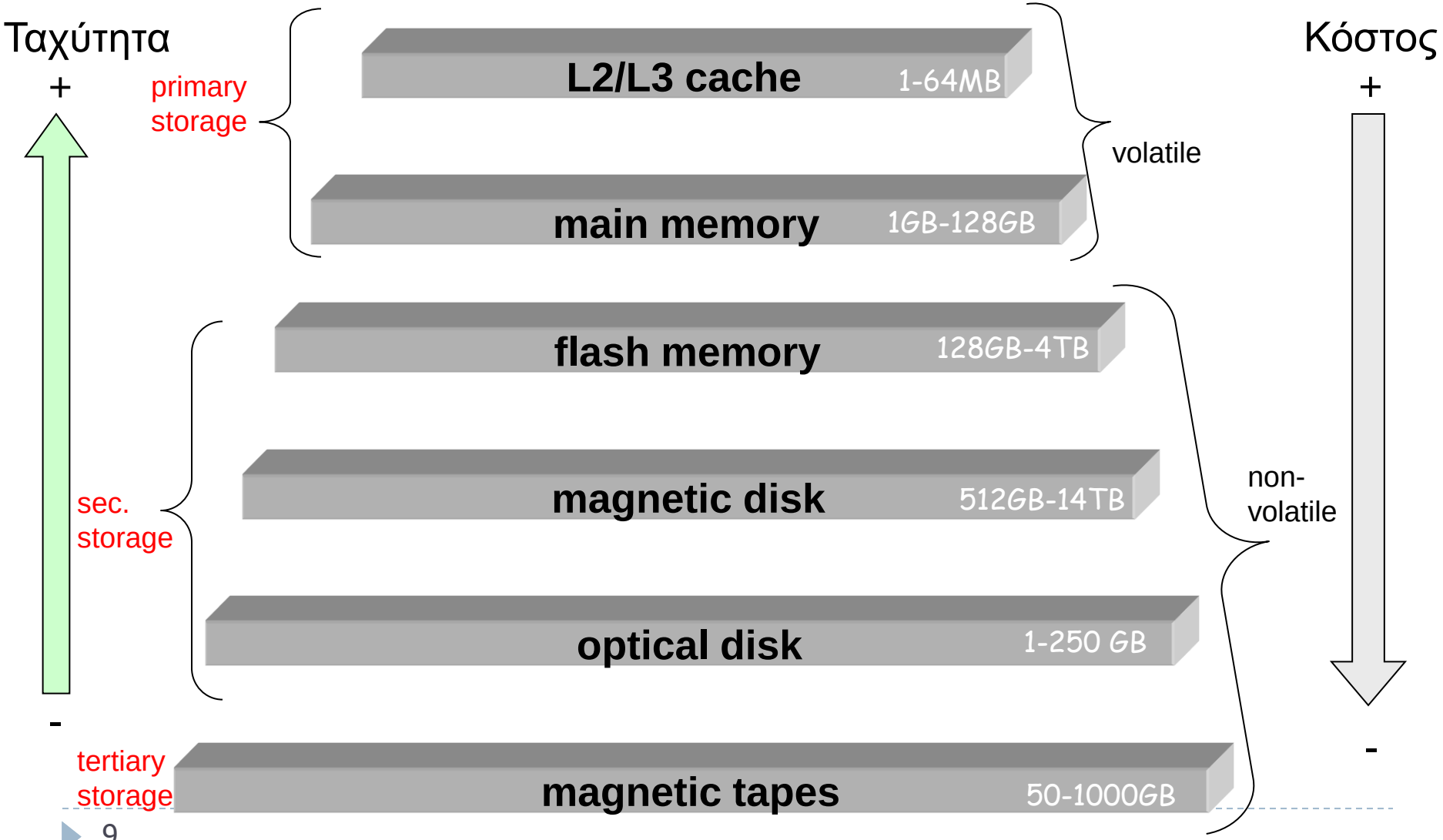
Ταξινόμηση Μνήμης

- Διαφορετικοί τρόποι κατηγοριοποίησης
  - Volatile (πρόσκαιρη) ←→ Non-volatile (μόνιμη)
- *Κύρια (Primary)*: fast, volatile
- *Δευτερεύουσα (secondary*): moderately fast, non-volatile
- *Ταινίες (Τριτογενής/Tertiary):* slow, non-volatile

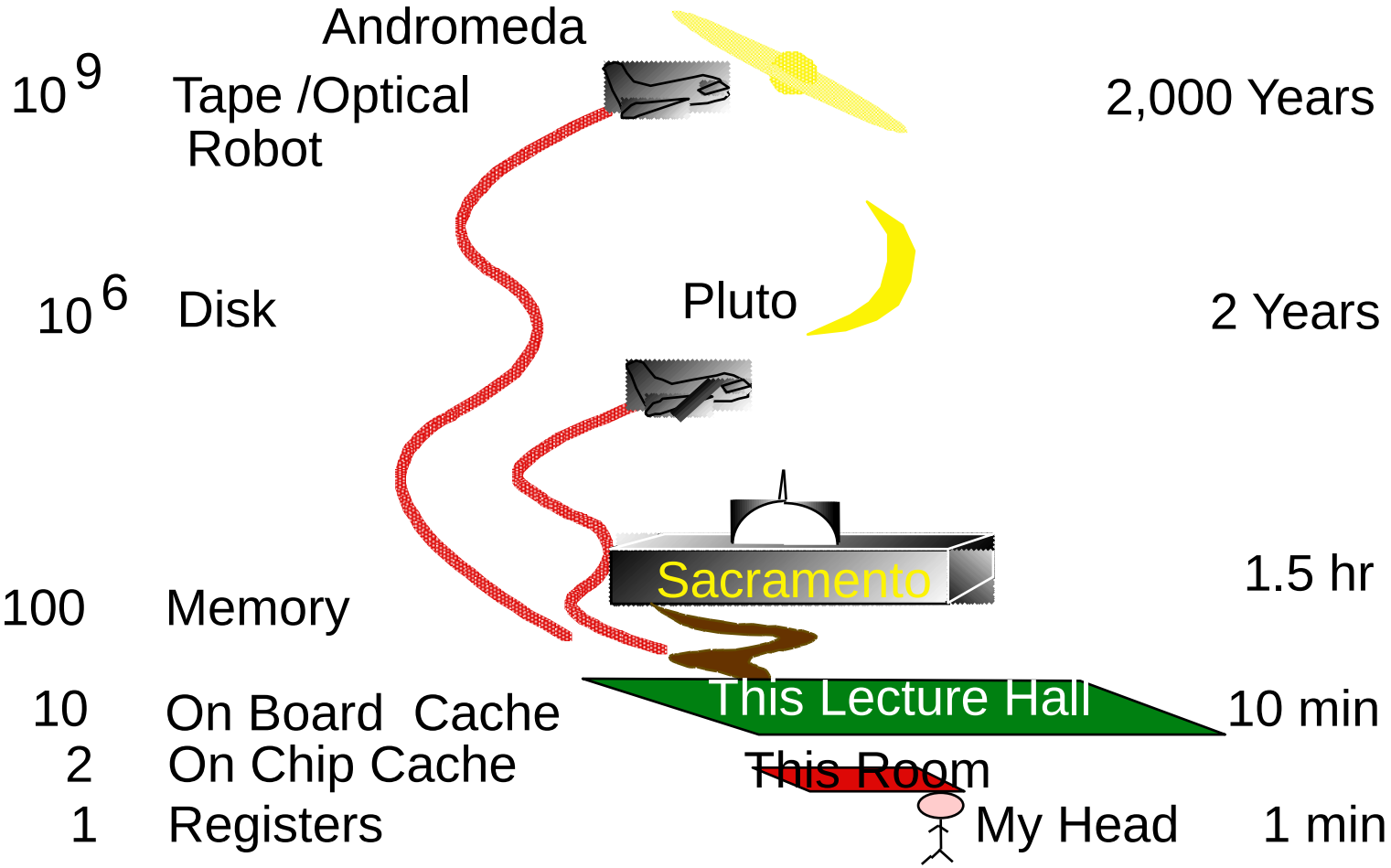
Volatile vs Non-Volatile

- Σημαντικό στοιχείο στη σχεδίαση ενός ΣΔΒΔ
  - *Επηρεάζει δύο από τα χαρακτηριστικά μίας ACID δοσοληψίας*
    - *Atomicity, Durability*
      - *E.g. what happens when parts of the data are in memory and parts in the disk and there is a failure?*
- Ακόμα και αν όλη η ΒΔ χωράει στη μνήμη, οι αλλαγές πρέπει να αποθηκευτούν σε κάποιο μόνιμο μέσο
  - Hard disk
  - RAM disks w/ battery
  - Flash memory

Ιεραρχία Μνήμης (sizes are outdated!)

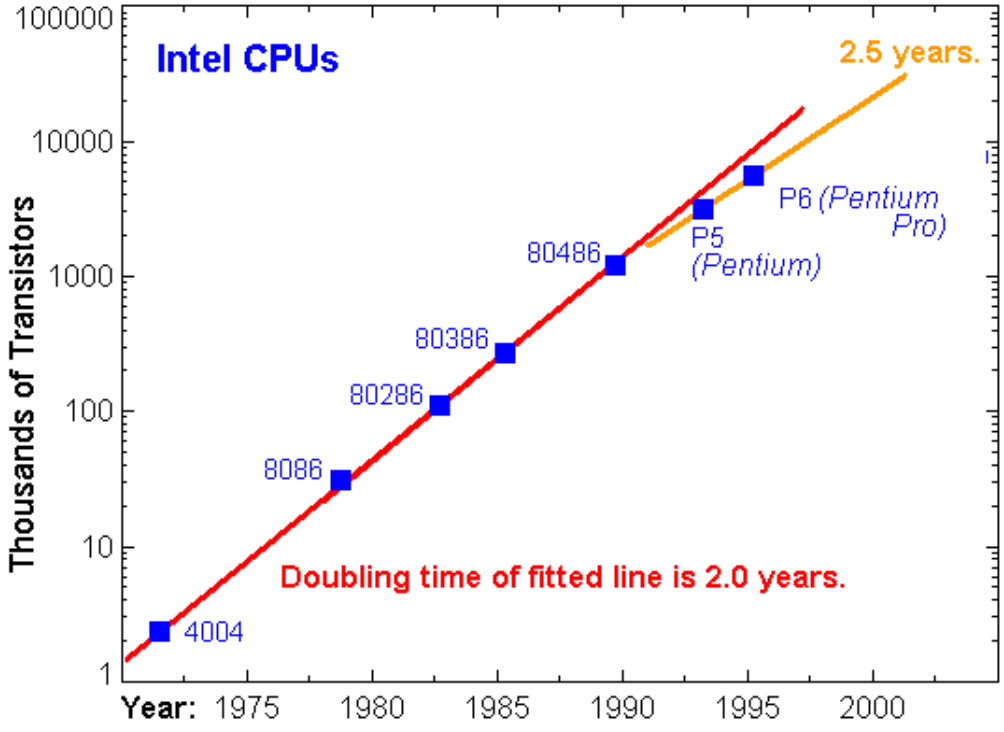


Jim Gray: How Far Away is the Data?



Moore's Law

- Gordon Moore (1965): Ο αριθμός των τρανζίστορ ανά τετραγωνικό εκατοστό διπλασιάζεται κάθε χρόνο



  - x2 τρανζίστορ / 18 μήνες

Αποτυχημένες Προβλέψεις

- Thomas Watson (builder of IBM, in 1943)
  - *"I think there is a world market for maybe five computers."*
- Ken Olsen (founder of Digital Equipment, in 1977)
  - *"There is no reason anyone would want a computer in their home."*

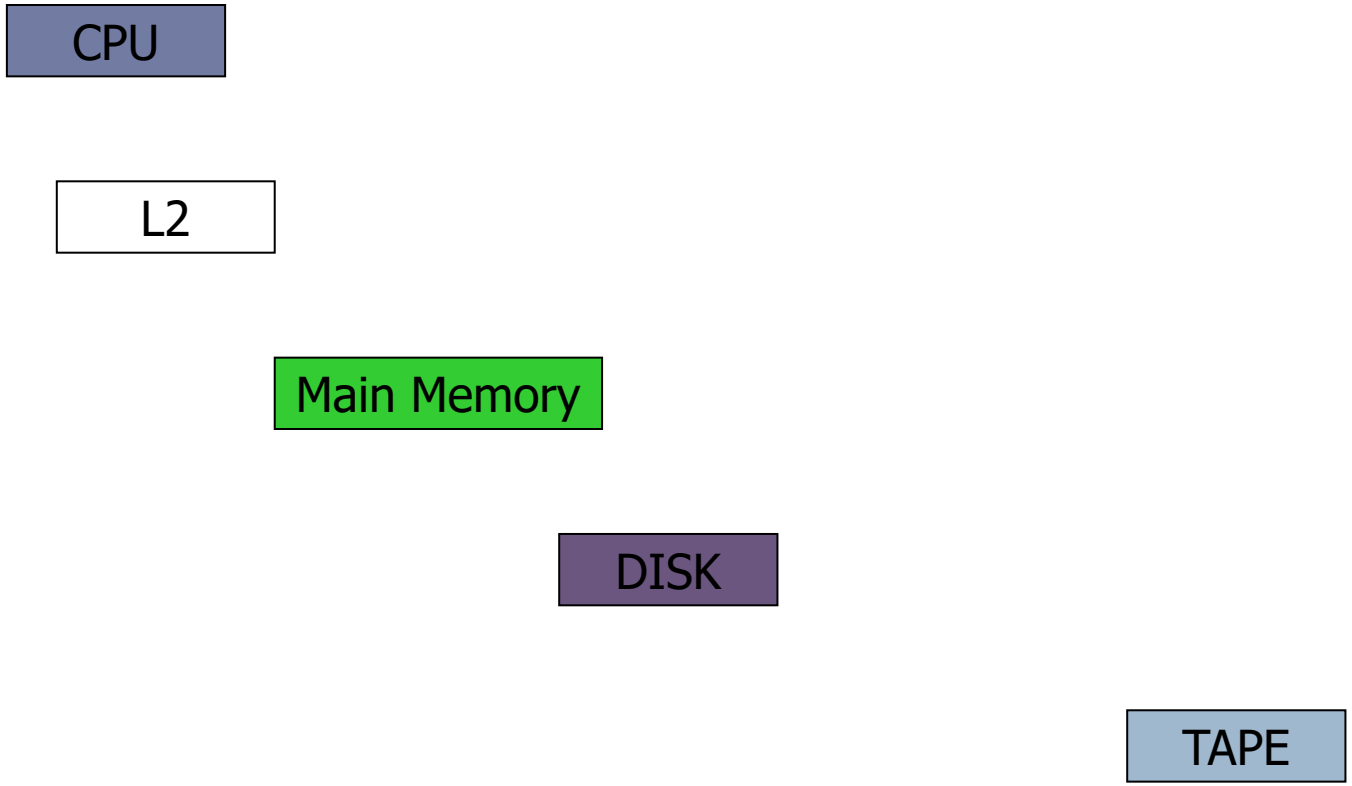
Moore's Law

- Χαρακτηριστικά που ακολουθούν το νόμο του Moore
  - Ταχύτητα επεξεργαστή
    - Νέα τάση: βελτίωση υπολογιστικής ισχύος / Watt
  - Αριθμός bits σε ένα chip
  - Αριθμός bytes σε ένα μαγνητικό μέσο
- Χαρακτηριστικά που **δεν** ακολουθούν το νόμο του Moore
  - Ταχύτητα προσπέλασης στην κύρια μνήμη
  - Ταχύτητα περιστροφής μαγνητικών δίσκων

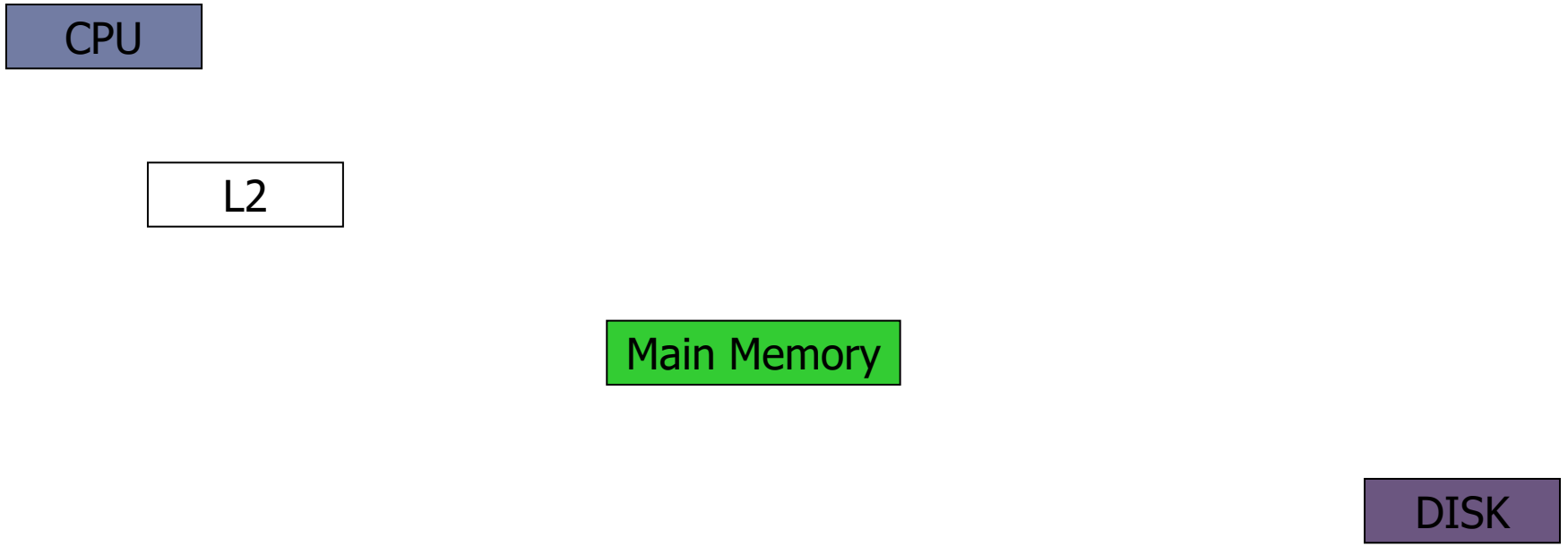
⇒ Καθυστέρηση (Latency) γίνεται σταδιακά μεγαλύτερη

⇒ Ο χρόνος για να μετακινήσουμε δεδομένα/εγγραφές ανάμεσα στα διάφορα επίπεδα της ιεραρχίας αυξάνει σημαντικά σε σχέση με το χρόνο επεξεργασίας στην CPU

Σήμερα

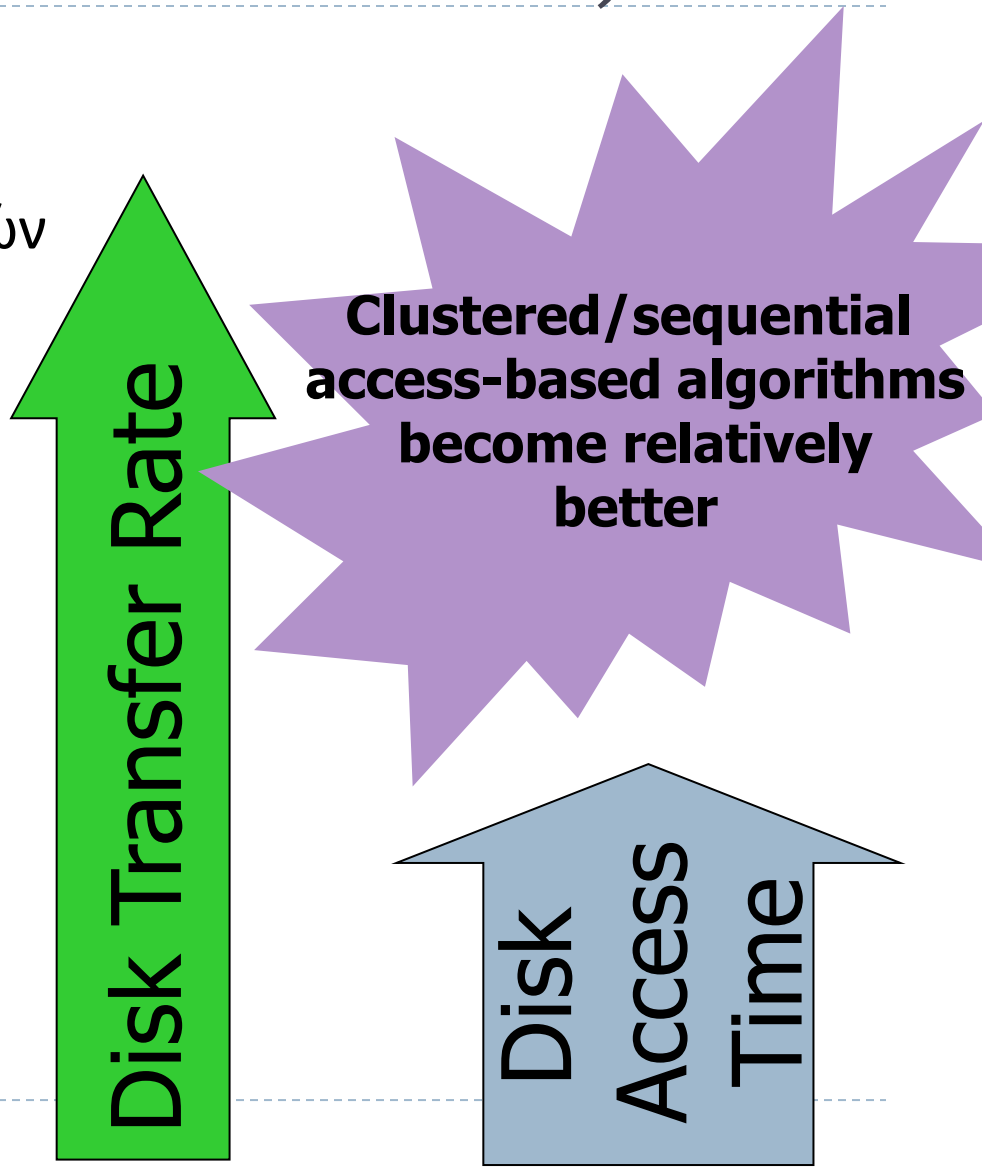


Σε λίγα χρόνια



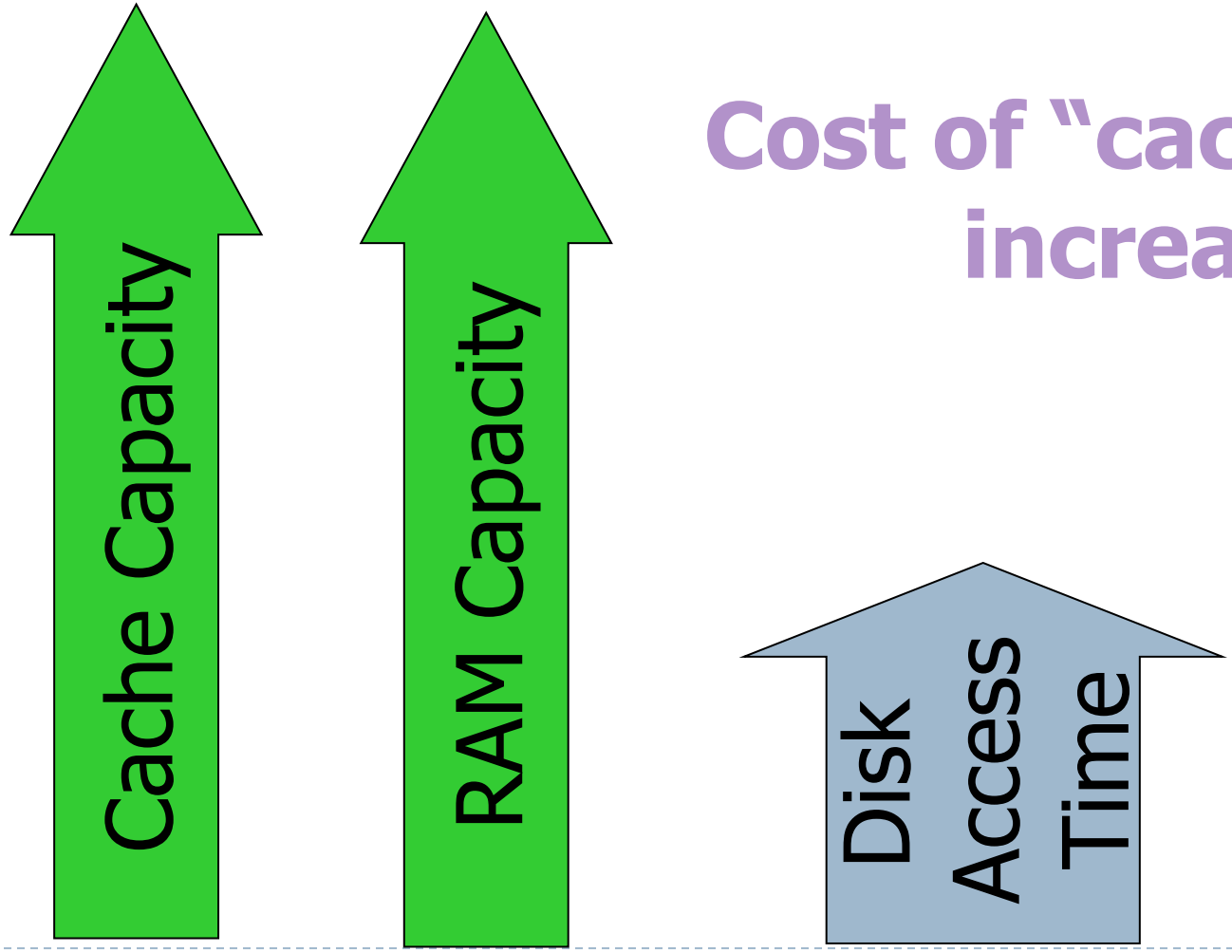
Moore's Law: συνέπειες

- Διαφορετικοί ρυθμοί βελτίωσης χαρακτηριστικών προσπέλασης
  - Ταχύτητα επεξεργαστή
  - Κόστος μνήμης/HDD/SSD
  - Ταχύτητα μνήμης
  - Ταχύτητα περιστροφής HDD
  - Ταχύτητα μεταφοράς
- …οδηγούν σε ανάγκη για αναθεώρηση

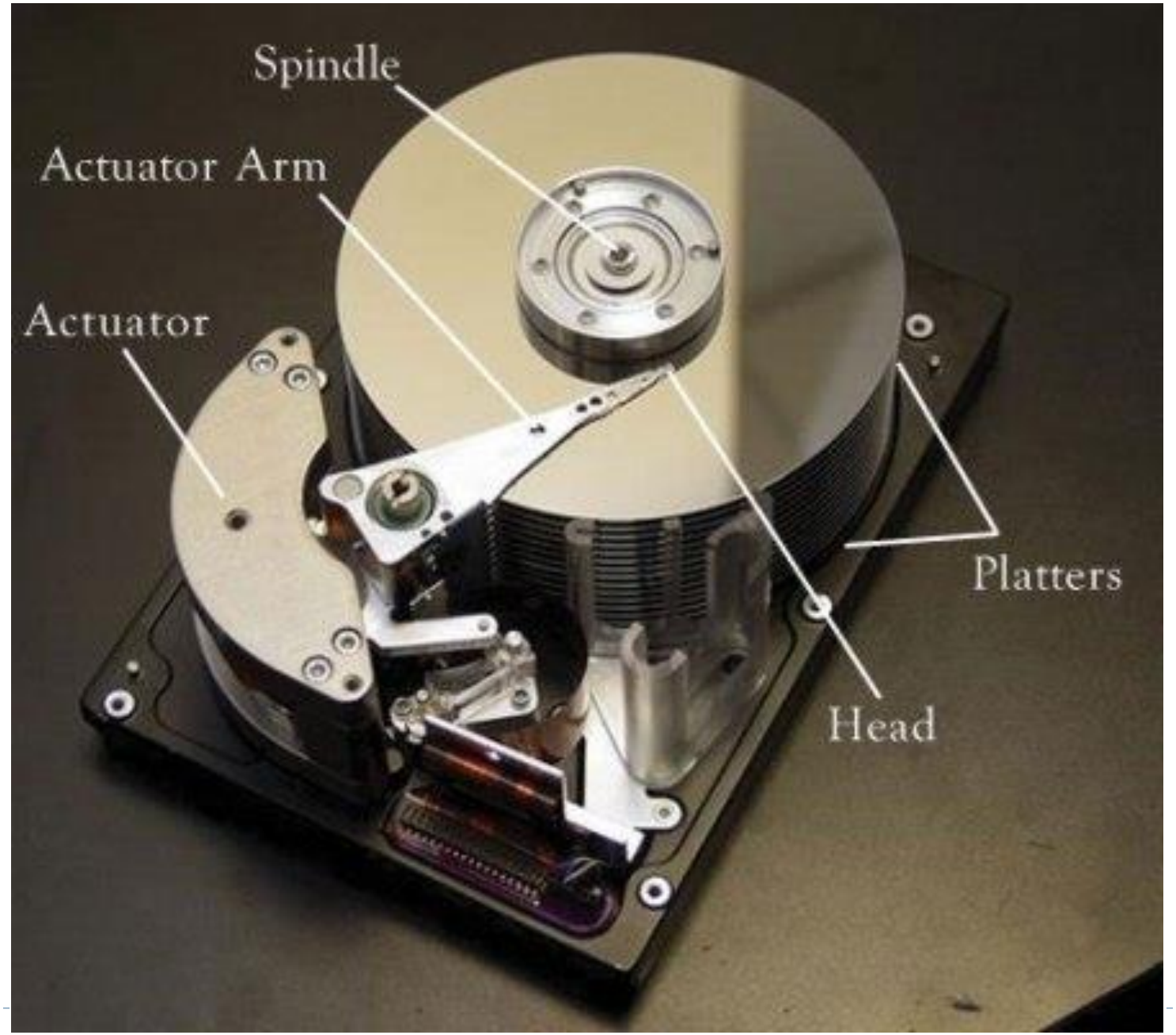


Το ίδιο φαινόμενο και στη RAM

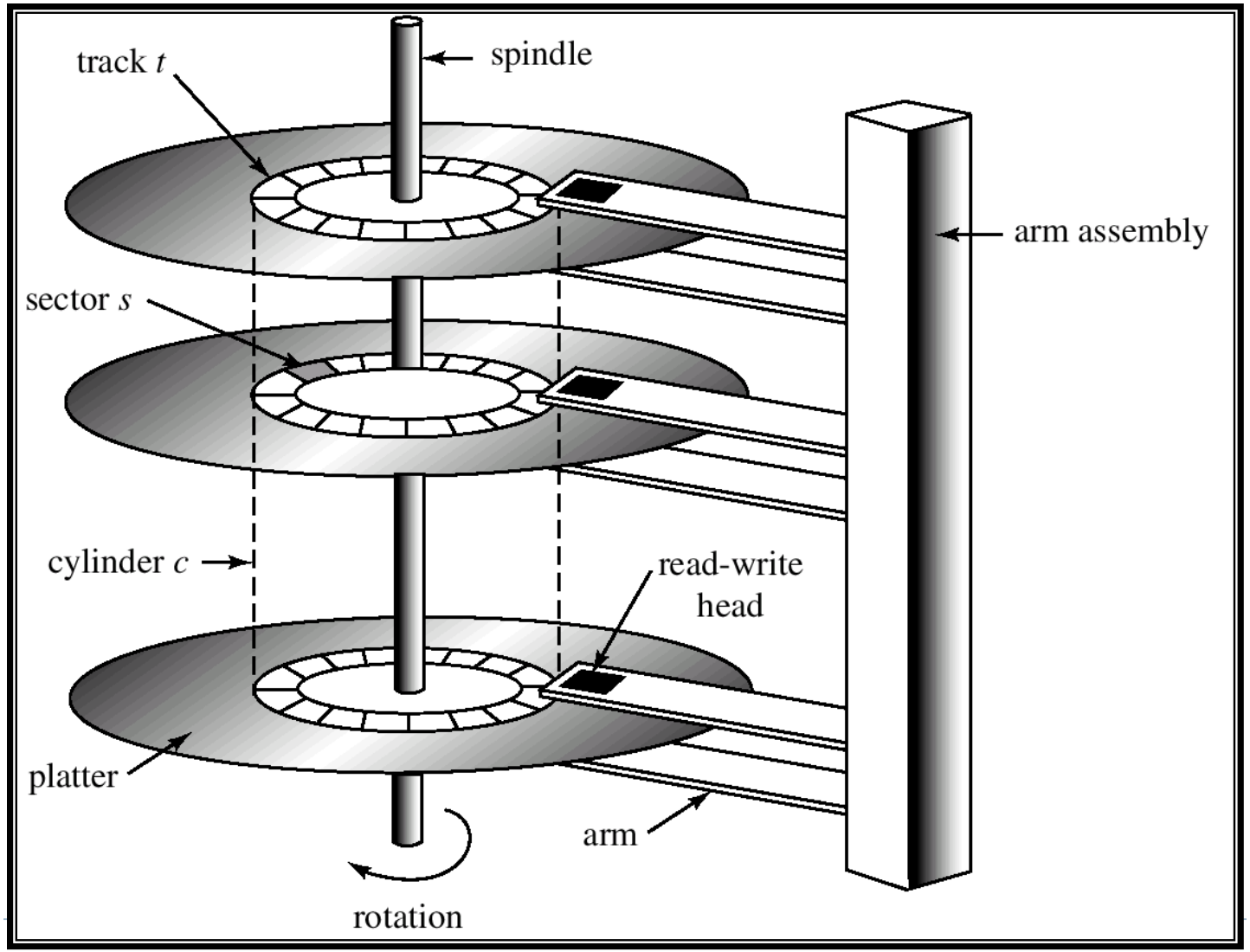
Data Locality Important



Cost of "cache-miss" increases


Spindle
Actuator Arm
Actuator
Platters
Head

Δομή Μαγνητικού Δίσκου

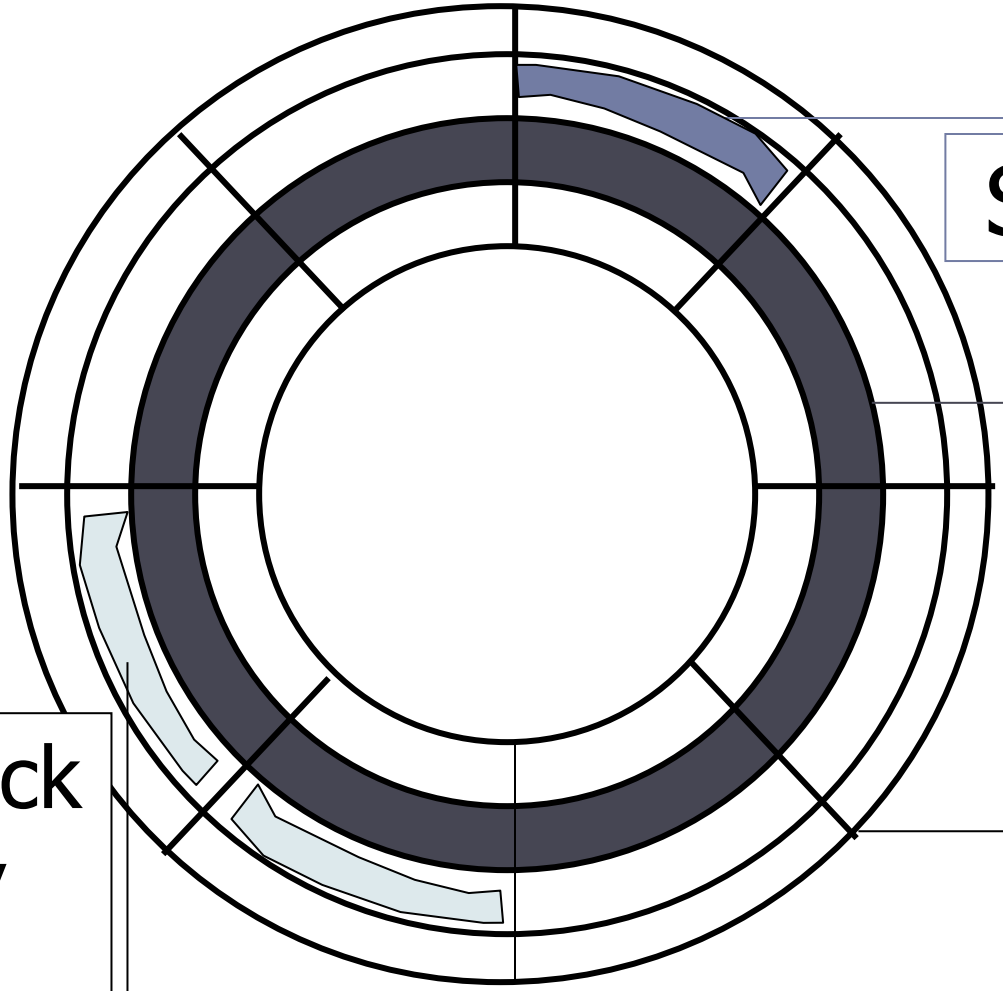


Ορολογία

- Πλατό/πλακέτα (plate)
- Επιφάνεια (surface)
- Ίχνη/άτρακτοι/τροχιές (tracks)
- Τομείς (sectors)
- Κεφαλή (head)
- Σελίδα (block/page)

*Συνήθως έχουμε διαφορετικό αριθμό sectors ανά track (περισσότερα στα εξωτερικά)

Απλοποιημένη Εικόνα*



Sector

Track

Gap

Logical Block (typically multiple sectors)

Παραδείγματα

Diameter:	1 inch $\rightarrow$ 15 inches
Cylinders:	100 $\rightarrow$ 40000
Surfaces:	1 (CDs) $\rightarrow$
	2 (floppies)
	6-10 (HD)
Sector Size:	512B $\rightarrow$ 54K
Capacity:	360KB (old floppy)
	12TB (newer HDD)

Physical Block Address

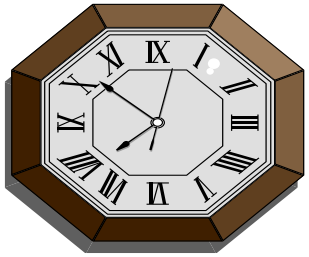
- Physical Device (ποιος δίσκος, σε ποιο controller)
  - Cylinder #
  - Surface # (or head#)
  - Sector

Λογική Δομή Δίσκου

- Οι εφαρμογές βλέπουν το δίσκο σαν ένα μονοδιάστατο πίνακα από λογικές σελίδες (logical blocks).
  - Οι εγγραφές διαβάζουν και γράφουν σελίδες.
- Τα logical blocks του μονοδιάστατου πίνακα αντιστοιχίζονται με φυσικές σελίδες (physical blocks) αποτελούμενες από 1 ή περισσότερα sectors.
  - Logical block addressing (LBA)

Disk Access Time

I want block X

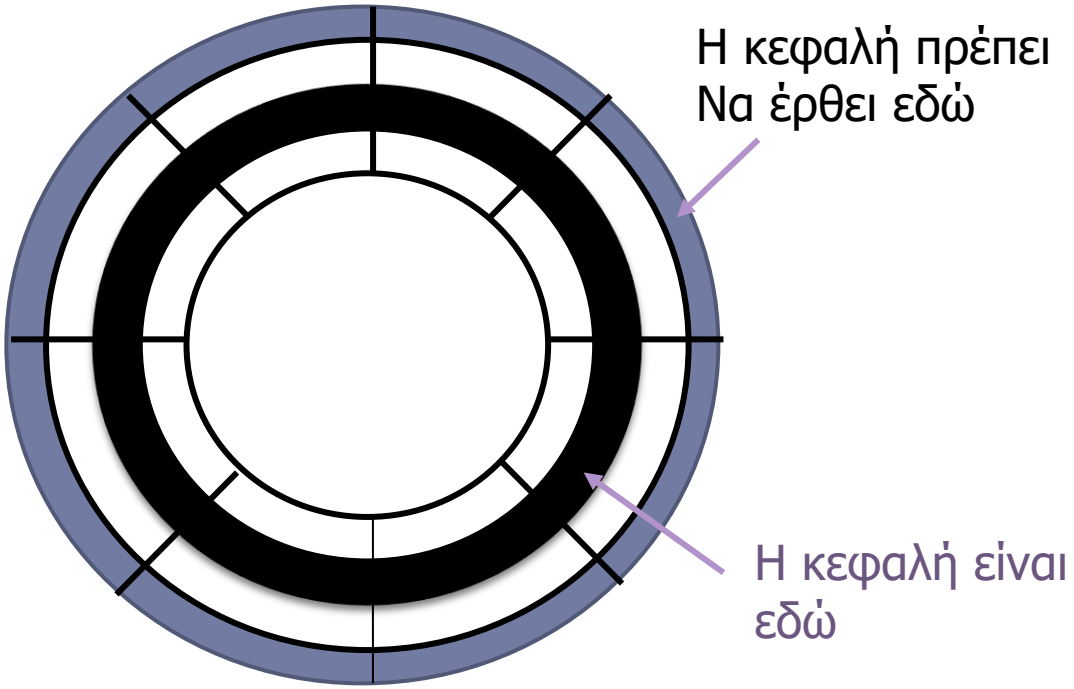


block X in memory

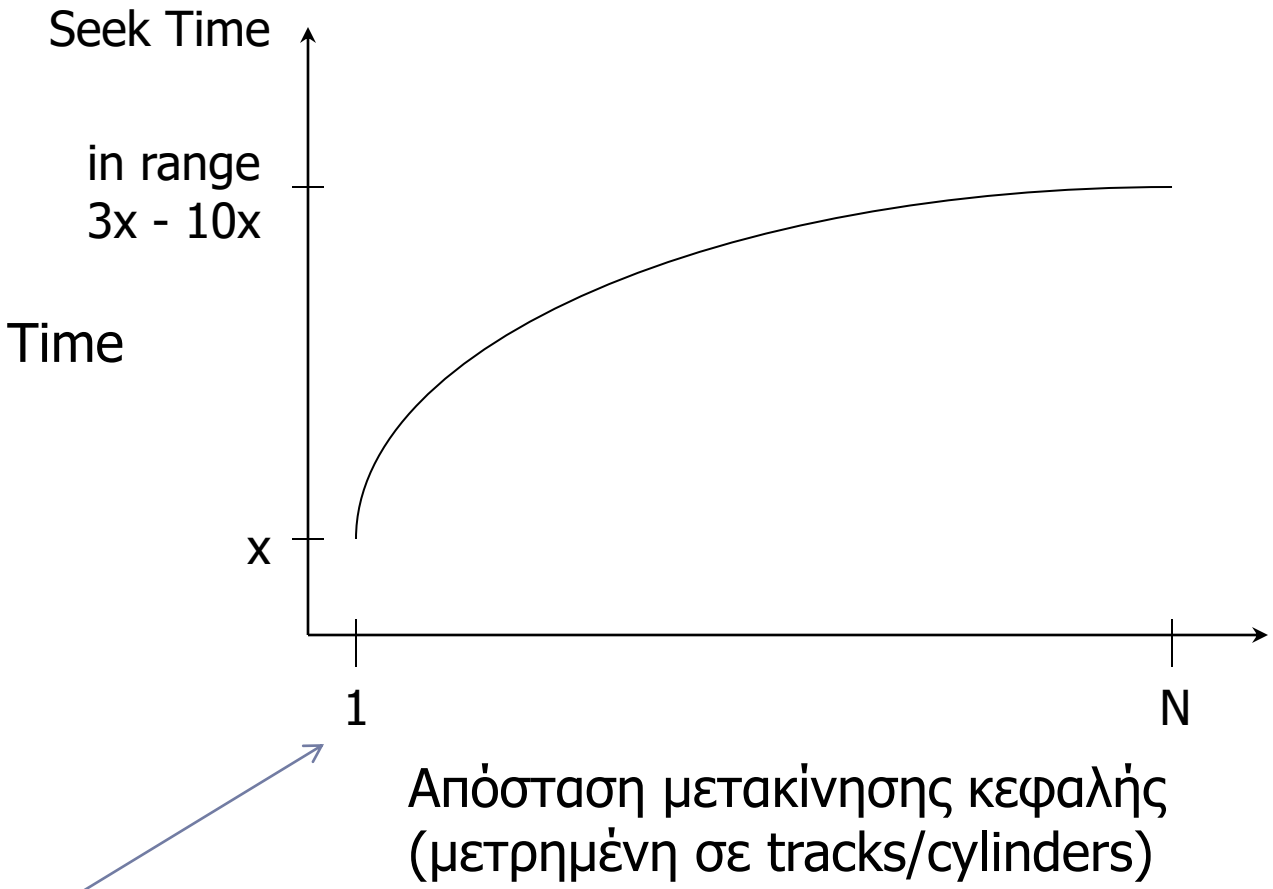
Time = Seek Time (locate track) +
Rotational Delay (locate sector)+
Transfer Time (fetch block) +
Other (disk controller, ...)

Χρόνος Αναζήτησης (Seek Delay/Time)

- Ο χρόνος αναζήτησης ενός δίσκου προσμετράει το χρόνο που απαιτείται για να μετακινηθεί η κεφαλή ανάγνωσης/εγγραφής μεταξύ των ιχνών.



Χρόνος Αναζήτησης σε συνάρτηση με απόσταση μετακίνησης των κεφαλών



Χρόνος εναλλαγής ίχνους/κυλίνδρου (~2msecs)

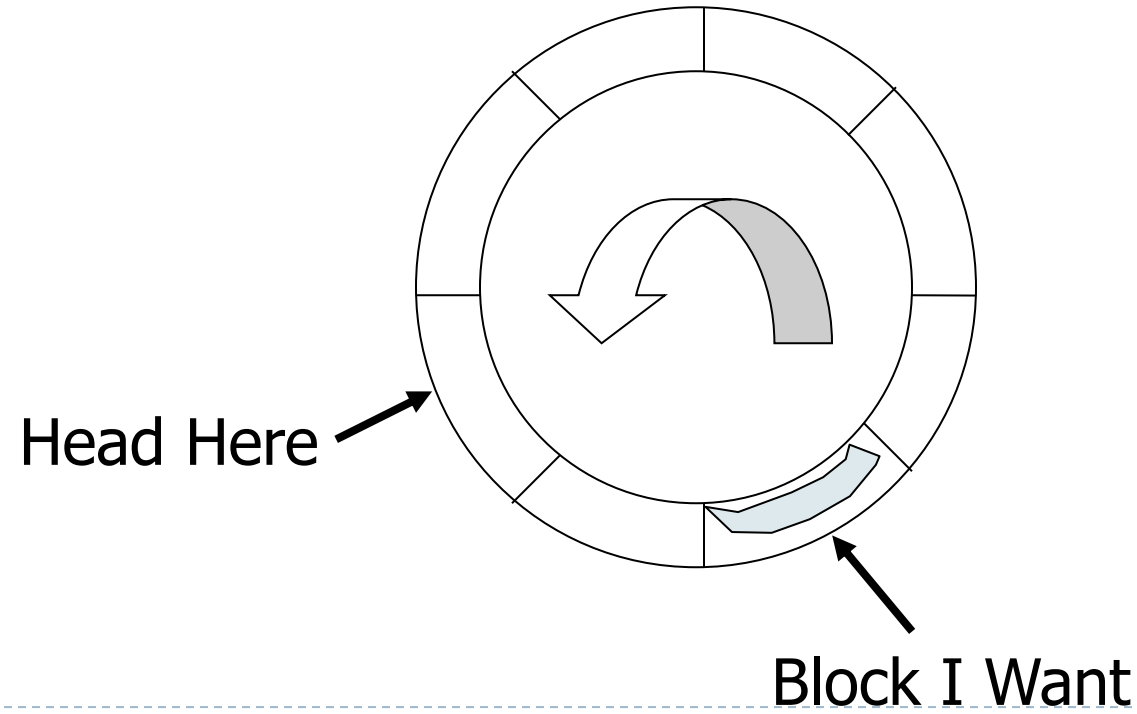
Average Random Seek Time

$$S = \frac{\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \text{SEEKTIME}(i \rightarrow j)}{N(N-1)}$$

"Typical" S: 8 ms $\rightarrow$ 40 ms

Λανθάνων χρόνος περιστροφής (Rotational Delay)

- **Λανθάνων χρόνος περιστροφής** είναι ο χρόνος, τον οποίο ο οδηγός πρέπει να περιμένει μέχρι ο σωστός τομέας να φτάσει κάτω από την κεφαλή ανάγνωσης/εγγραφής.



Average Rotational Delay

- Κατά μέσο όρο περιμένω μισή περιστροφή
- R = ο χρόνος για 1/2 περιστροφή

- Έστω δίσκος 7200 RPM

$$\frac{60 \text{ secs}}{R} \qquad \frac{7200 \text{ περιστροφές}}{½ \text{ περιστροφή}}$$

$$R = 30 / 7200 = 4{,}17\text{msecs}$$

Ταχύτητα Μεταφοράς (Transfer Time) (1 block)

- Ο χρόνος που απαιτείται για να «περάσουν» τα sectors του block κάτω από τις κεφαλές
  - Εξαρτάται από την ταχύτητα περιστροφής και το μέγεθος του block (σε sectors)
- Παράδειγμα:
  - 7200 RPM, 128 sectors/track, 2 sectors/block.
  - Σε 1 περιστροφή(=60/7200secs) διαβάζω 128 sectors
  - Για να διαβάσω 1 block=2 διαδοχικά sectors θέλω

    (2/128) * (60/7200) = 0,13 msecs
  - Συγκριτικά:
    - seek time ~20msecs, rotational delay ~4msecs

Άλλες καθυστερήσεις

- Χρόνος CPU για στείλει την εντολή για I/O
- Χρόνος επεξεργασίας αιτήματος στον disk controller
- Χρόνος μεταφοράς στη μνήμη μέσω του διαύλου

"Typical" Value: 0

Παράδειγμα

- 1 επιφάνεια
- Rotation speed 7200rpm
- 16384 tracks
- 128 sectors/track
- 4096 bytes/sector
- 4 sectors/block (16KB/block)
- SEEKTIME ($i \rightarrow j$) = [1000 + |j-i|] μs

- Υπολογίστε ελάχιστο, μέγιστο και μέσο χρόνο για την ανάγνωση ενός block.

Υπολογισμός Ελαχίστου Χρόνου

- Η κεφαλή είναι στην αρχή του 1ου sector του block που θέλουμε να διαβάσουμε
  - Υπολογίζω μόνο το transfer time
- 1 block = 4 sectors είναι τα 4/128 ενός track
- 1 περιστροφή παίρνει 60/7200=8,33ms
- Transfer time = 8,33 * 4 /128 = 0,26ms
  - Εξήγηση: σε μια πλήρη περιστροφή διαβάσω 128 sectors, άρα για να διαβάσω 4 συνεχόμενα χρειάζομαι το 4/128 του χρόνου
- Total time (min) = 0,26ms

Υπολογισμός Μέγιστου Χρόνου

- Υποθέτω μέγιστη μετακίνηση κεφαλής (πχ από το ποιο εσωτερικό στο ποιο εξωτερικό track)
  - Seek time = 1000+(16384-1)μs=17,38ms
- Μετά το seek, η κεφαλή μόλις έχασε την αρχή του block που ψάχνω
  - Full rotational delay =60/7200 = 8,33ms
- Transfer time (όπως πριν) = 0,26ms
- Total time(max)=17,38+8,33+0,26=25,97ms

~100 x min-time!!!

Μέσος χρόνος

- Seek time = AVERAGE(1000+|i-j|)=…=6,46ms
- Rotational delay: μισή περιστροφή = ½*60/7200secs=4,17ms
- Transfer Time 0,26ms
- Σύνολο = 10,89ms

Ελάχιστος Χρόνος	0,25ms
Μέσος Χρόνος	10,89ms
Μέγιστος Χρόνος	25,97ms

- Είδαμε: τυχαία προσπέλαση
- Τι γίνεται αν θέλω να διαβάσω το αμέσως επόμενο block;

Αν γίνει σωστά…

Time to get next block = 0,26ms + Negligible

- skip gap
- once in a while, next cylinder

Παρατήρηση

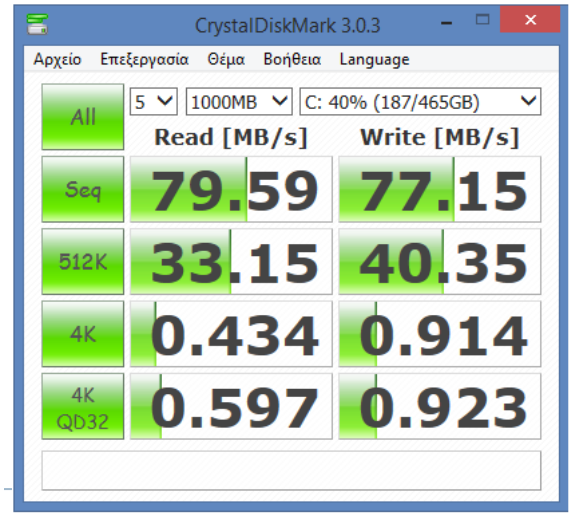
Rule of Thumb

Random I/O: Ακριβό
Sequential I/O: **Πολύ** Φθηνότερο

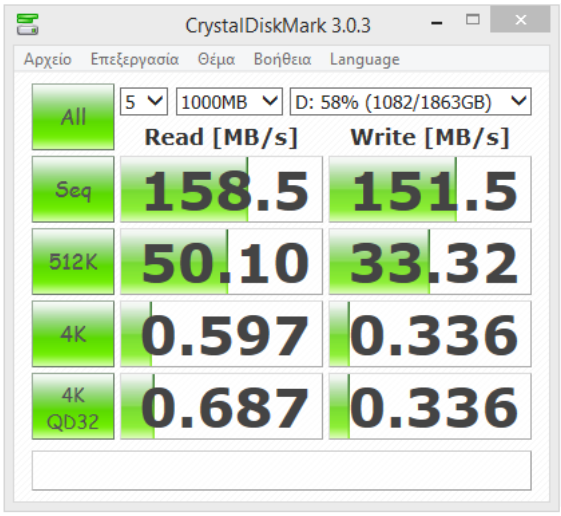
- Χρόνος για να διαβάσω 1Block
  - Random I/O: 11 ms. (avg)
  - Sequential I/O: <1ms.

Σειριακή ταχύτητα ανάγνωσης

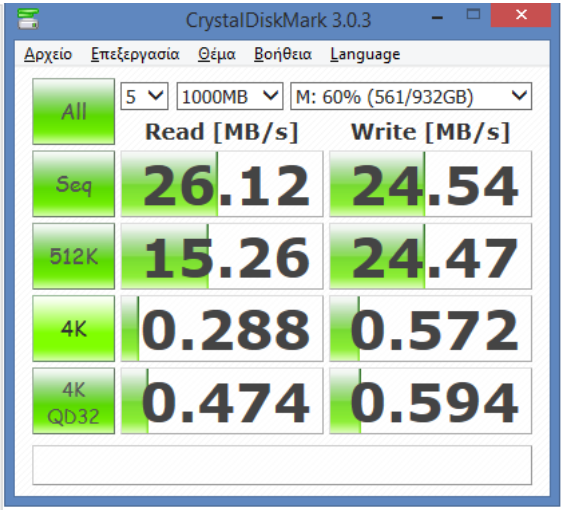
- Τι γίνεται αν διαβάζω διαδοχικά sectors στο ίδιο track;
  - Διαδοχικά track στον ίδιο κύλινδρο
- 128 sectors/track * 4KB/sector σε 8,33ms/track = 512KB/8.33ms = 60MB/sec!!!
  - Μοντέρνοι δίσκοι (PC) 150-250MB/sec
- Random I/O (avg): 11ms/4KB → 0,33MB/sec



hdd



hdd



external hdd

Εγγραφή

- Το κόστος εγγραφής είναι παρόμοιο με το κόστος ανάγνωσης
- …εκτός και αν θέλουμε να επαληθεύσουμε, οπότε περιμένουμε μία πλήρη περιστροφή για να διαβάσουμε ότι γράψαμε

Ενημέρωση σελίδας (block)

- Για να αλλάξω το περιεχόμενο μίας σελίδας (block)
  - Διαβάσω τη σελίδα στη μνήμη
  - Κάνω ότι αλλαγές χρειάζονται στο αντίγραφο στη μνήμη
  - Γράφω τη σελίδα πίσω στο δίσκο
- Προσοχή: τα παραπάνω γίνονται ακόμα και θέλω να αλλάζω την τιμή ενός bit!

Παράδειγμα- Megatron 747 Disk

- 3600 RPM
- 1 surface
- 16 MB usable capacity (16×2^{20})
- $128=2^7$ cylinders
- 1 block=1 sector=1 KB
- 10% κενό (gaps) ανάμεσα στα sectors
- Average Seek time = 25 ms
- Seek time για να πάω στον επόμενο κύλινδρο/ίχνος = 5 ms.

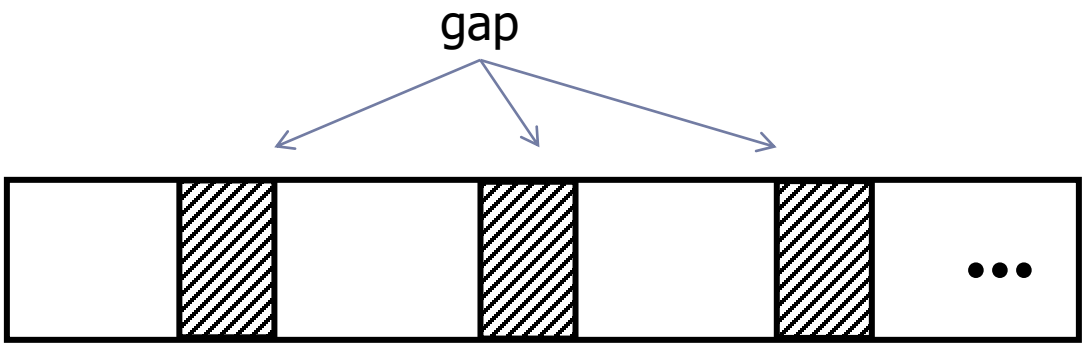
Παρατηρήσεις

- 1 επιφάνεια→ο κύλινδρος ταυτίζεται με το ίχνος (track)
  - Για το παράδειγμα αυτό μόνο!
- Χωρητικότητα = 16 MB = $(2^{20})16 = 2^{24}$
- bytes/cyl = 16 MB/128 = $2^{24}/2^{7} = 2^{17}$ = 128 KB
  = 16 * 1024KB / 128 = 16 * 8 KB = 128 KB
- blocks/cyl = 128 KB / 1 KB = 128

- Δηλαδή κάθε track έχει 128 sectors του 1KB

3600 RPM: 60/3600 = 16,66ms για 1 περιστροφή

One track:



- In a full rotation (16.66ms)
  - Time over useful data: 16,66 * 90%=14,99 ms.
  - Time over gaps: 16,66 * 10% = 1,66 ms.
- Transfer time 1 block = 14,99/128=0,117 ms.
- Trans. time 1 block+gap=16,66/128=0,13 ms.

T_1 = Time to read one random block (1KB Block)

T1 = seek + rotational delay + transfer time

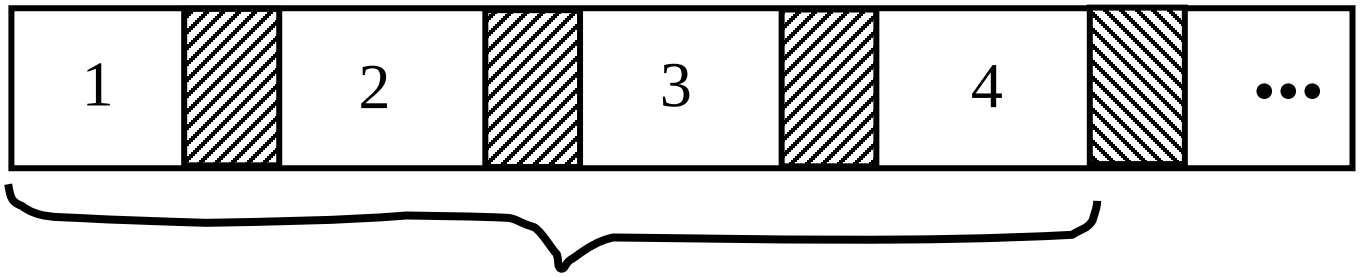
= 25 + (16,66/2) + 0,117 = **33,45 ms**

υποθέτω μισή περιστροφή

Διαχωρισμός block/σελίδας

- Ως τώρα υποθέταμε ότι η «σελίδα» που χρησιμοποιεί το O/S και οι εφαρμογές ταυτίζεται με 1 block στο δίσκο
- Στην πράξη το O/S (ή το ΣΔΒΔ) μπορεί να χρησιμοποιεί το δικό του «μέγεθος σελίδας»
  - Μπορεί να αλλάζει ανάλογα με τη δομή αποθήκευσης (σχέση, ευρετήριο)
  - Windows NTFS: 4KB
  - SQL Server: 8KB
  - Σε εφαρμογές μεγάλων δεδομένων (big data) συνήθως είναι πολύ μεγαλύτερο (Hadoop HDFS: 128MB)
    - Λιγότερο bookkeeping
    - Σειριακή ανάγνωση είναι ο κανόνας

Διάβασμα σελίδας 4KB (four sectors) από το δίσκο



One sector+ gap

One sector, no gap

$T_4 = 25 + (16,66/2) + (0,117) \times 1$

$+ (0,130) \times 3 = 33,83$ ms

[Compare to $T_1 = 33,45$ ms]

Τι γίνεται αν 1 σελίδα=1 track=128KB;

T_T = Time to read a full track

(υποθέτω για το παράδειγμα ότι μπορώ να αρχίσω το διάβασμα από οποιοδήποτε sector του track)

$T_T = 25 + (0,130/2) + 16,66 - 0,013 = 41,72$ ms

to get to start of next sector

do not have to read last gap

Ας συγκρίνουμε το κόστος για να διαβάσω 1 σελίδα

- Read 1KB page : 33,45ms
- Read 4KB page : 33,83ms
- Read 128KB page : 41,73ms

- Γιατί να μη μεγαλώσω και άλλο το μέγεθος της σελίδας?

Σωστό Μέγεθος Σελίδας ΣΔΒΔ?

- Μεγάλη σελίδα -> διαμοιράζεται το κόστος ανάγνωσης ανάμεσα σε πολλές εγγραφές
- Λιγότερα αιτήματα στον δίσκο για να διαβάσω μία μεγάλη σχέση

ΑΛΛΑ

- Ίσως διαβάζω και πολλές άλλες εγγραφές που δε χρειάζομαι (και ο χρόνος ανεβαίνει, έστω και λίγο)
  - Επίσης η μεγάλη σελίδα όταν έρθει στη μνήμη καταλαμβάνει περισσότερο χώρο
- Τι συμβαίνει κατά την εκτέλεση της παρακάτω SQL επερώτησης?
  - **update Account set Balance=Balance+100 where AcountNumber=12345**

Τι πρέπει να σκεφτώ

- Update-heavy workload vs Read mostly ?
  - Transactions (e-shop orders, bank transfers)
  - Analytical queries (compute avg account balance per municipality)
  - Bulk processing (e.g. MapReduce)
- Μικρές – μεγάλες σχέσεις?
- Clustering εγγραφών/locality of access
- Ευρετήρια

- SQL Server: 8KB (fixed)
- Oracle: διαφορετικά μεγέθη σελίδας

A note on HDD technology*

- The dimensions of the head are impressive. With a width of less than a hundred nanometers and a thickness of about ten, it ies above the platter at a speed of up to 15,000 RPM, at a height that is the equivalent of 40 atoms. If you start multiplying these infinitesimally small numbers, you begin to get an idea of their significance.

- Consider this little comparison: if the read/write head were a Boeing 747, and the hard-disk platter were the surface of the Earth
  - The head would fly at Mach 800
  - At less than one centimeter from the ground
  - And count every blade of grass
  - Making fewer than 10 unrecoverable counting errors in an area equivalent to all of Ireland

*source: Matthieu Lamelot, Tom's Hardware

Coping with errors, disk failures

- Error correcting codes (single disk)
- Disk arrays (multiple disks)

Checksums For Failure Detection

- Idea: add one parity bit so that #1s is **always even**
  - Block A: 01101000:**1** (odd # of 1's → parity =1)
  - Block B: 11101110:**0** (even # of 1's → parity =0)

- Suppose due to an error the first bit in block A changes from 0 → 1
  - Block A': 11101000:1
  - Easy to detect, number of #1s with parity is odd (5)

- But suppose: Block A flips 2 bits
  - Block A": 01000000:1 (also has odd # of 1's)

  → **Can only detect an odd number of errors**

How to compute parity bit

- Modulo-2 sums = XOR
  - Modulo-2: %2

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

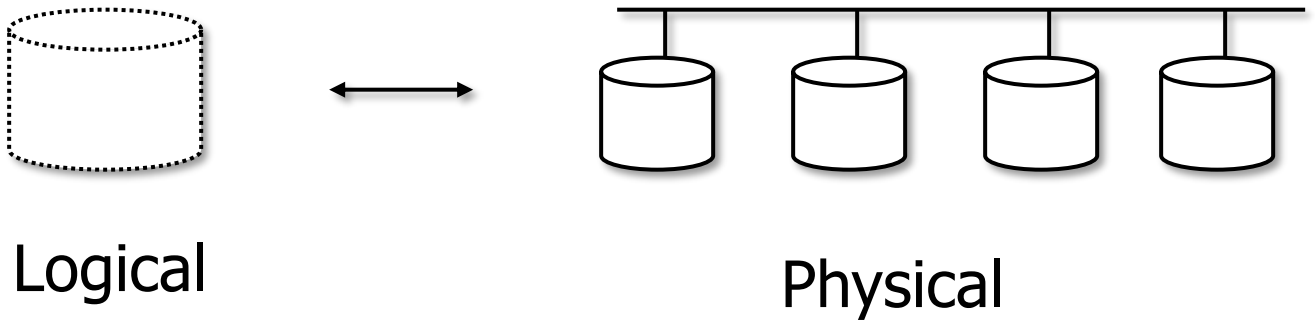
https://www.build-electronic-circuits.com/

- Example: compute parity bit for 01101000
  - Sum ones: 1+1+1 = 3
  - 3 modulo 2 = 1 (if the sum is odd → 1, otherwise 0)
    - Notice that 0 XOR 1 XOR 1 XOR 0 XOR 1 XOR 0 XOR 0 XOR 0 = 1
  - Thus, parity bit = 1

- 01100101 ?
  - 1+1+1+1 = 4. Since 4 %2 = 0→ parity bit = 0
    - 0 XOR 1 XOR 1 XOR 0 XOR 0 XOR 1 XOR 0 XOR 1 = ?

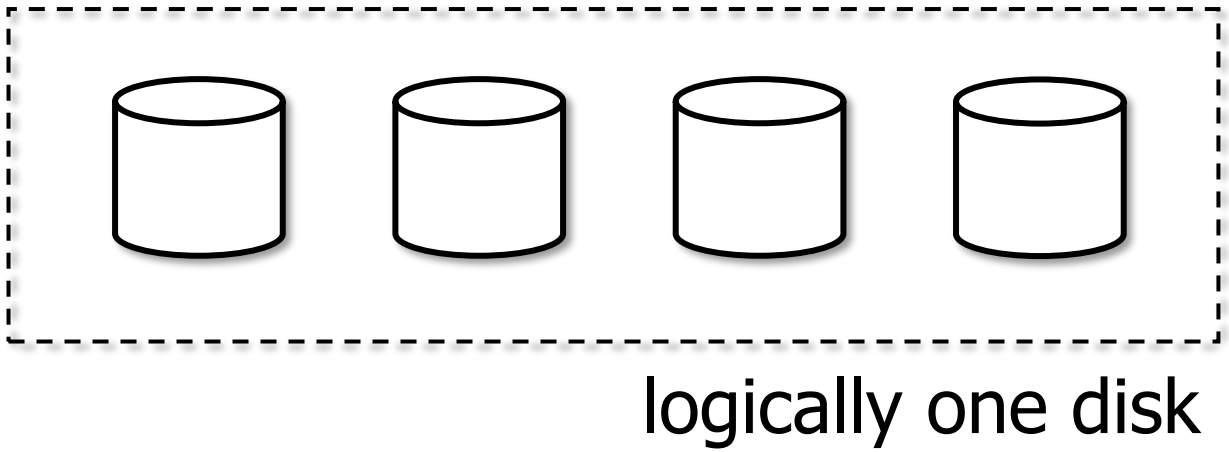
At what level do we cope?

- Single Disk
  - e.g., Error Correcting Codes
- Multiple Disks: Disk Arrays/RAID systems



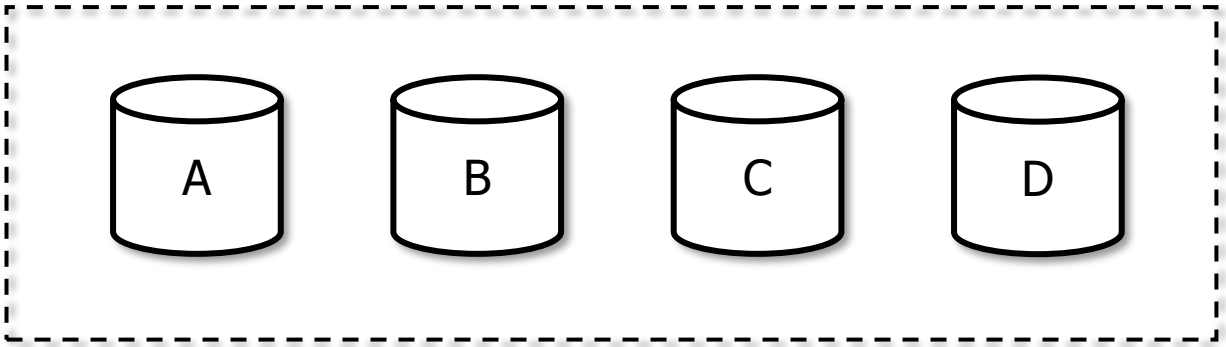
Disk Arrays

- RAIDs (Redundant Arrays of Inexpensive/Independent Disks)
  - (various flavors/levels)



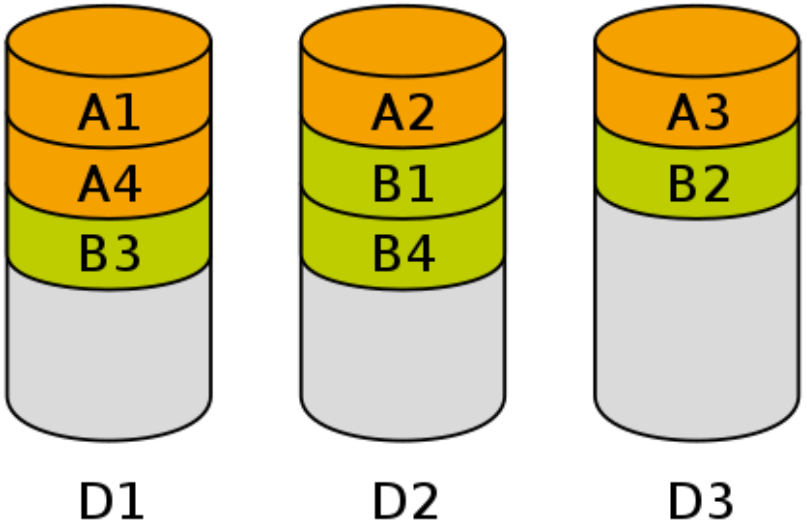
Disk Arrays

- RAID Level 0 (Striping)
  - Distribute the contents of each file among *all* disks
  - Utilize all disks during Reads/Writes
  - Retain full capacity of all disks
  - Failure of any disk causes the entire RAID 0 volume to be lost
  - Possible in NAS devices with 2 disks BUT RISKY



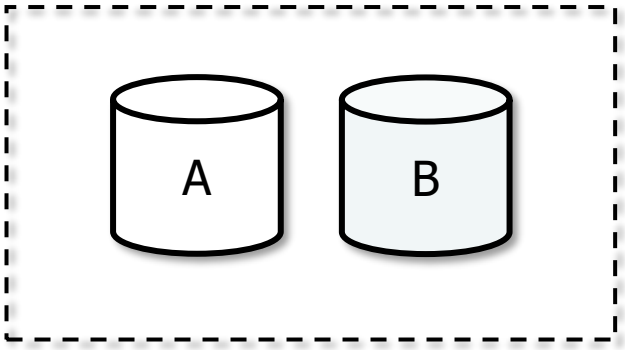
Raid 0 example (from Wikipedia)

- Content of files A, B is spread across multiple devices
  - Segments Ai, Bi are called "stripes"
  - Stipe size (typical examples): One page (4KB) or some multiple of it



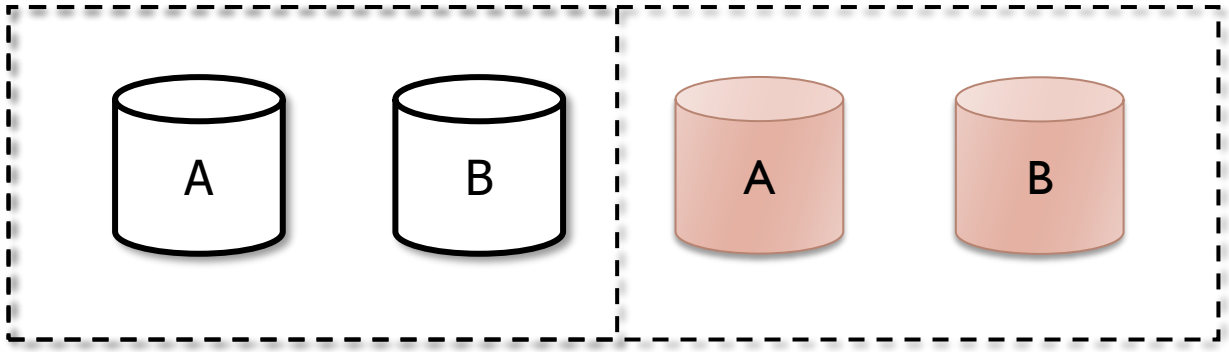
Disk Arrays

- RAID Level 1 (Mirroring)
  - Keep exact copy of data on redundant disk(s)
  - Common in NAS devices with 2 disks



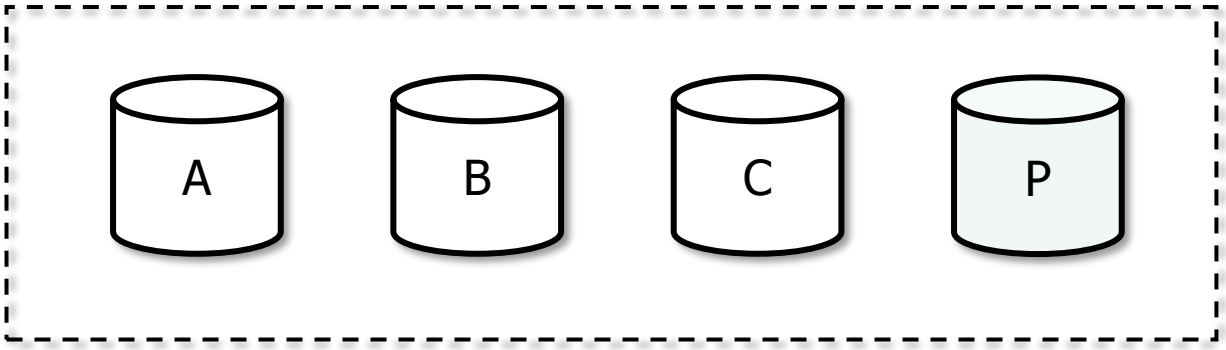
Disk Arrays

- RAID Level 0+1 (Nested RAID)
  - Stripe across A+B (level 0), mirror both disks (level 1)
  - 50% of capacity is lost due to mirroring (level 1)

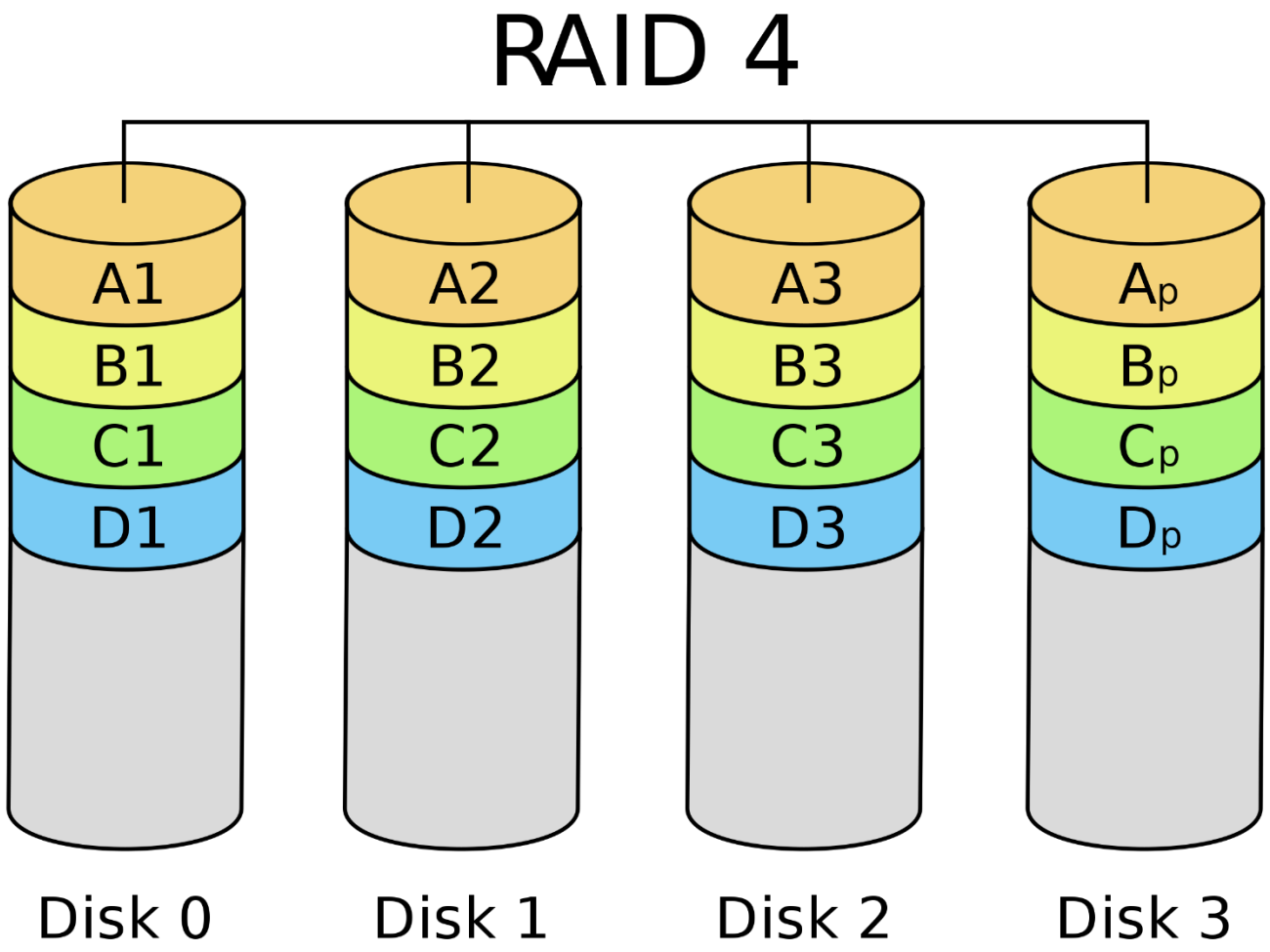


Disk Arrays

- RAID Level 4 (striping + parity disk)
  - Striping unit is one block
  - Keep only one redundant disk
  - Store parity blocks on redundant disk

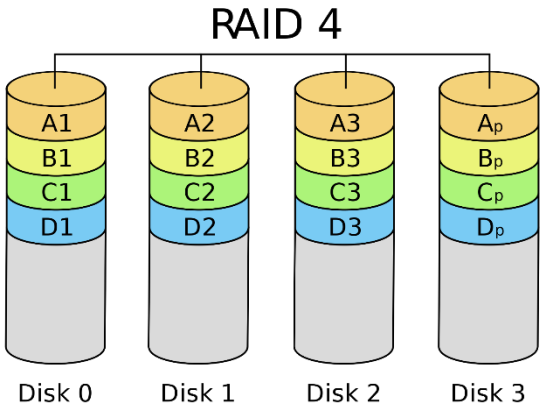


RAID 4 - Wikipedia



Parity Blocks & Modulo-2 Sums

- Have an array of 3 data disks
  - Disk 1, block 1 (A1): 11110000
  - Disk 2, block 1 (A2): 10101010
  - Disk 3, block 1 (A3): 00111000
- … and 1 parity disk
  - Disk 4, block 1 (Ap): 01100010



- Note: - Sum over each column is always an even # of 1's
  - Mod-2 sum can recover any missing single row (e.g., a logical block in one of the disks)

Using Mod-2 Sums for Error Recovery

- Suppose we have:
  - Disk 1, block 1: **11110000**
  - Disk 2, block 1: **???????? (disk is gone or block is damaged)**
  - Disk 3, block 1: **00111000**
  - Disk 4, block 1: **01100010** (Parity)

- Mod-2 sums (or XOR) for block 1 over disks 1,3,4:
  → Disk 2, block 1: **11110000 XOR 00111000 XOR 00111000 = 10101010**

Block update RAID 4

- Have an array of 3 data disks + 1 parity
- Application updates block 1 on disk 2
  - Disk 1, block 1: **11110000**
  - Disk 2, block 1: **10101010 (old) ➔ 11101000 (new)**
  - Disk 3, block 1: **00111000**
  - Disk 4, block 1: **01100010 (old parity)**
- **Naïve re-computation of parity block (slow):**
  - Read block 1 from disks 1,3, recompute parity using new data on disk 2, write back to disk 4
  - New parity block:
    - 11110000 XOR 11101000 XOR 00111000 = 00100000
  - Notice that a single-block update operation involves all disks (slow)

Block update RAID 4

- Have an array of 3 data disks + 1 parity
- Application updates block 1 on disk 2
  - Disk 1, block 1: **11110000**
  - Disk 2, block 1: **10101010 (old) ➔ 11101000 (new)**
  - Disk 3, block 1: **00111000**
  - Disk 4, block 1: **01100010 (old parity)**
- **Alternative technique:**
  - Mod-2 Sum between old block, old parity, new block
  - **10101010 XOR 01100010 XOR 11101000**

    Force old parity to forget old block

  - **Rule: (new parity) = (old data) XOR (old parity) XOR (new data)**

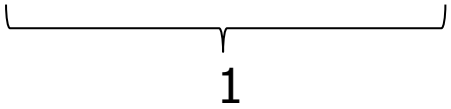
Properties of x XOR y

- Result is 1 if input bits differ, 0 otherwise

- 0101 XOR 0110 = 0011

- 0011 XOR 0110 = 0101

and



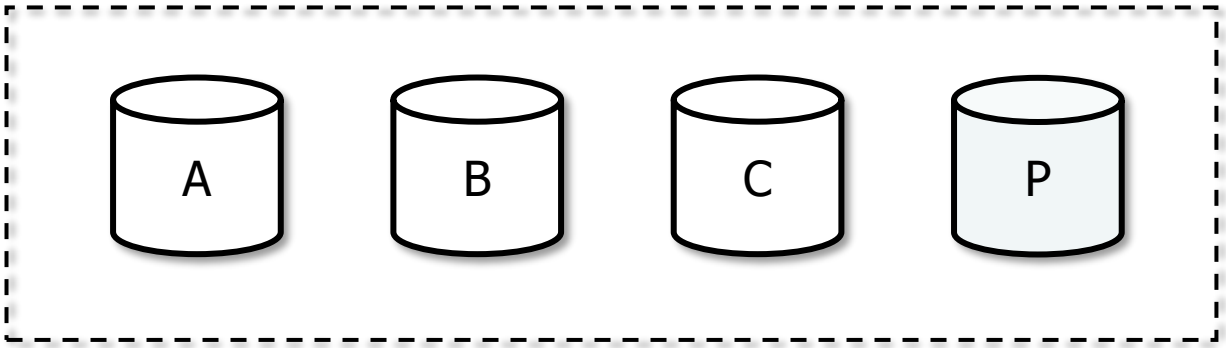
- 0011 XOR 0101 = 0110

Quiz

- (A XOR B XOR C) XOR C XOR A = ?
- (A XOR B XOR C XOR D) XOR C = ?

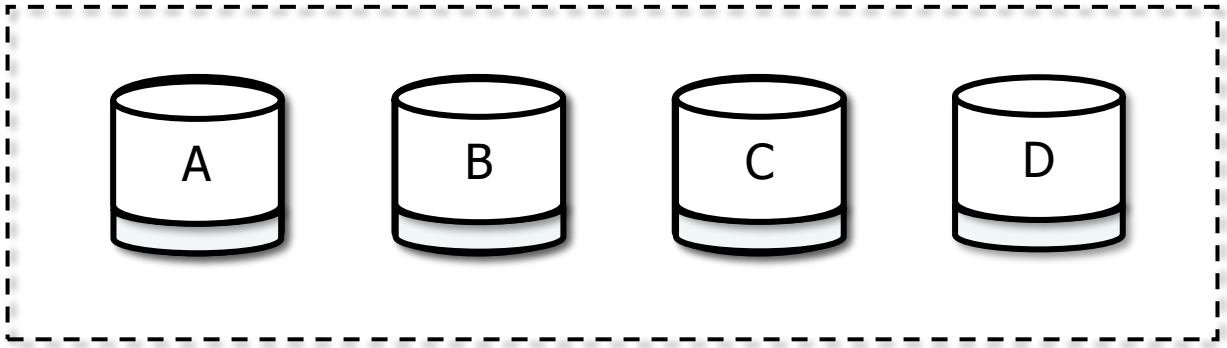
RAID 4

- How many drives participate in concurrent reads?
  - App data is fetched from disks A,B,C
  - 25% of read bandwidth is lost, in the example bellow
- How many drives participate in a write operation?
  - Parity disk penalty + overhead of reading old status



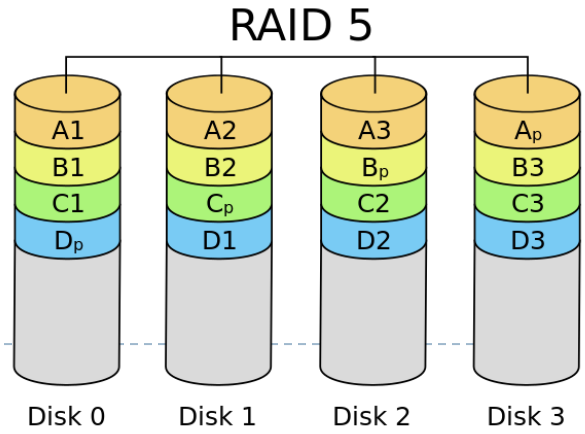
Disk Arrays

- RAID Level 5 (Striping + distributed parity)
  - Like level 4, but balanced read & write load



→ Parity partition on each disk

From Wikipedia:



Ο κανόνας των 5 λεπτών (Jim Gray 1986)

- Προσπαθεί να ποσοτικοποιήσει το κέρδος του να τοποθετώ εγγραφές που προσπελαύνονται συχνά στη μνήμη.
- Χρησιμοποιεί δεδομένα όπως τα μέγεθος, κόστος, ταχύτητα μνήμης και δίσκου για να απαντήσει στο ερώτημα πόση μνήμη χρειάζομαι σε ένα ΣΔΒΔ.

Παράδειγμα (από το 1986)

- Ένας HDD Tandem κοστίζει 15K$ και προσφέρει 15 random accesses/sec. Κόστος controller & I/O interface άλλα 15K$
  - Έστω A=Κόστος/αριθμός προσπελάσεων/δευτερόλεπτο
    - A=30000/15 = 2000$/accesses/sec
  - Έστω ότι μία εγγραφή αναζητείται από την εφαρμογή κάθε (Read Interval) RI δευτερόλεπτα
    - 1/RI accesses / sec
    - A = cost/accesses/sec
  - Κόστος σχετιζόμενο με την ανάκτηση της εγγραφής από το δίσκο A*(1/RI) = A/RI
    - Αυτό το κόστος μπορώ να το αποφύγω αν εγγραφές που χρησιμοποιώ συχνά τις διατηρώ στην κύρια μνήμη

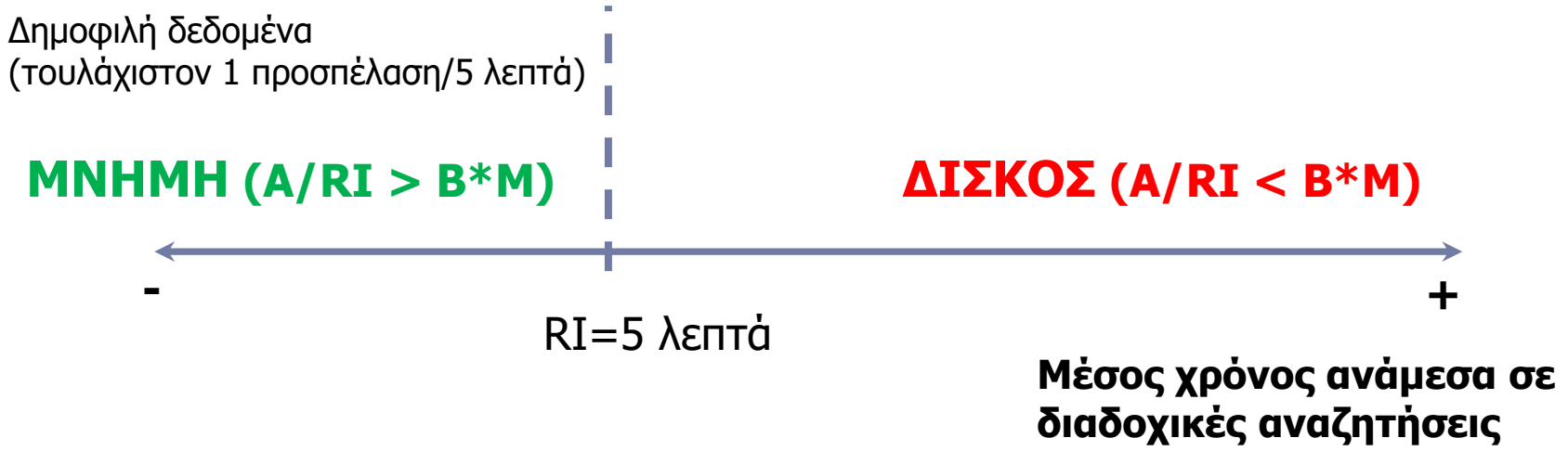
Κόστος Μνήμης (1986)

- 1MB μνήμης κοστίζει $5000 !
  - Κόστος/byte = M = 5000/1024/1024
- Έστω μία εγγραφή μεγέθους B (πχ B = 1KB = 1024Bytes)
- Κόστος αποθήκευσης της εγγραφής στη μνήμη
  - B*M
- Κόστος ανάκτησης από το δίσκο A/RI
- Ισορροπία όταν τα 2 κόστη είναι ίδια
- A/RI = B*M → RI = A/(M*B)
- Το 1986
  - RI = 2000/(5000/1024/1024 * 1024) = 409 secs ~ 5 λεπτά

Συσχετισμοί

A/RI ~ B*M

- Για δεδομένα που αναζητούνται συχνότερα από ανά 5 λεπτά το αριστερό μέλος (κόστος προσπέλασης δίσκου) είναι μεγαλύτερο → αποθήκευση στη μνήμη
- Για δεδομένα που προσπελαύνονται λιγότερο συχνά, το κόστος αποθήκευσης στη μνήμη είναι μεγαλύτερο → αποθήκευση στο δίσκο



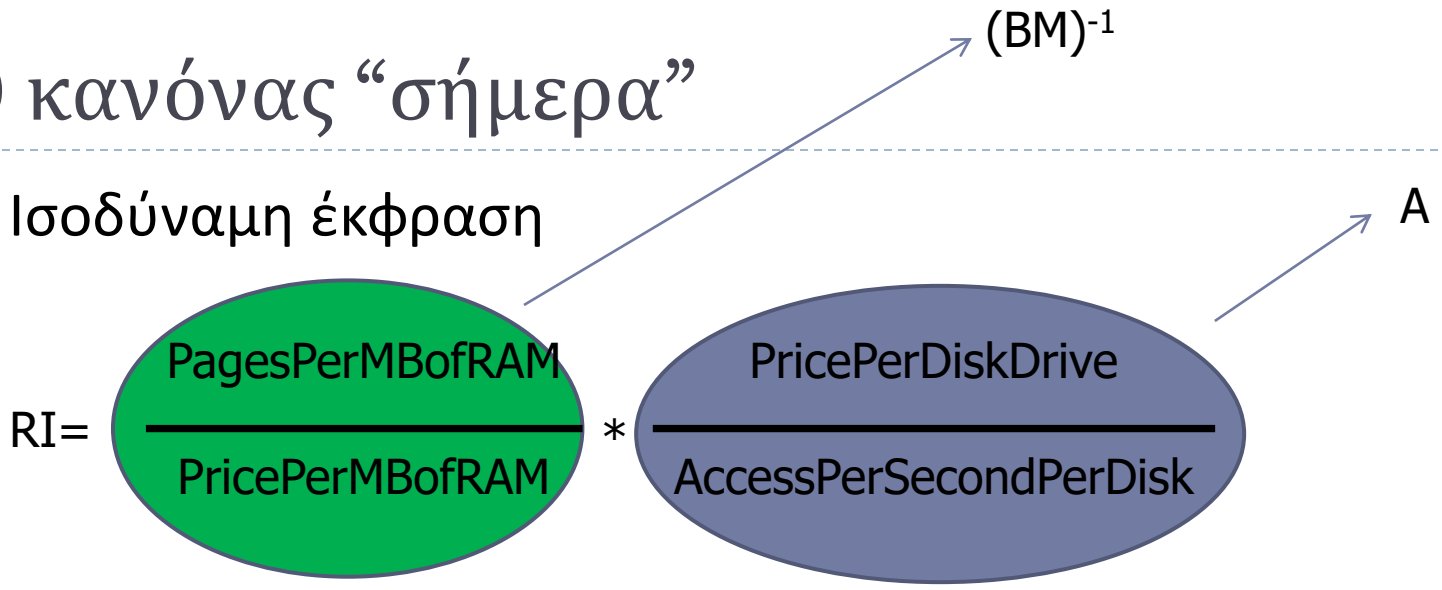
Κανόνας των 5 λεπτών

Δεδομένα που αναζητούνται κάθε 5 λεπτά ή συχνότερα πρέπει να βρίσκονται στη μνήμη.

- Τι έχει αλλάξει σήμερα?
  - Μνήμη σημαντικά ποιο φθηνή
  - Δίσκοι (HDD) ποιο φθηνοί αλλά μικρή βελτίωση στα accesses/sec
  - SSD: χιλιάδες accesses/sec αλλά αρκετά ποιο ακριβοί από HDD ($/bytes)

Ο κανόνας "σήμερα"

- Ισοδύναμη έκφραση



- 40 ευρώ 4GB μνήμη, 8KB μέγεθος σελίδας
- PricePerMBofRAM =40/(4*1024)
- PagesPerMBofRAM = 1024/8=128
- PricePerDiskDrive = 200 ευρώ (1TB disk+controller)
- AccessPerSecondPerDisk = 64 a/s/disk
- RI = 11 hours!
  - Κράτα τα records που διαβάζεις κάθε 11 ώρες ή συχνότερα στη μνήμη

1986 vs present!

Present:

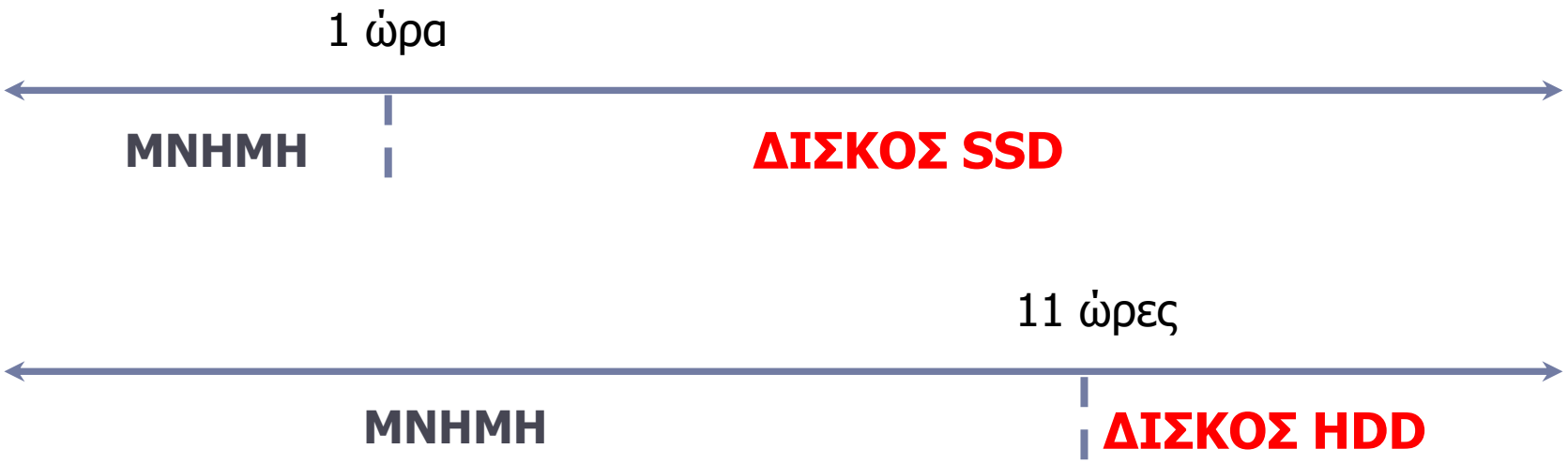
1986:

RAM vs SSD (outdated)

- 40 ευρώ 4GB μνήμη, 8KB μέγεθος σελίδας
- PricePerMBofRAM =40/(4*1024)
- PagesPerMBofRAM = 1024/8=128
- PricePerSSD = 500 ευρώ για 256GB
- AccessPerSecondPerDisk = 2000 a/s/disk
- RI = 1 hour
- Άρα αν χρησιμοποιώ SSD χρειάζομαι το ~1/10 της μνήμης (υποθέτοντας ομοιόμορφη κατανομή…) σε σχέση με ένα HDD

1986 vs present (outdated)!

Present:



1986:

SSD vs HDD

- Αν κατεβούμε 1 σκαλί την ιεραρχία μνήμης τότε μπορούμε να συγκρίνουμε το κέρδος από ένα SSD αν μπει σαν cache(=μνήμη) μπροστά από ένα HDD
- Στον προηγούμενο τύπο αντικατέστησε τα δεδομένα για RAM με τα δεδομένα για τους SSD
- PricePerMBofSSD =500/(256*1024)
- PagesPerMBofSSD = 1024/8=128
- PricePerHDD = 200 ευρώ
- AccessPerSecondPerHHD = 64 a/s/disk
- RI = …. = 58,3 ώρες !
  - Άρα μάλλον συμφέρει να βάλουμε ένα μεγάλο μέρος του dataset (ή όλο!) σε SSD

Optimizations: Prefetching

- Prefetching = λήψη δεδομένων πριν αυτά χρειαστούν
- Locality of access = δεδομένα, στο χρόνο καλούνται από διαδοχικές ή κοντινές περιοχές στο δίσκο.
- Παράδειγμα: table/relation scan
  - Προϋποθέτει ότι έχουμε μεριμνήσει οι σελίδες της σχέσης να βρίσκονται «κοντά» πχ στο ίδιο track, κύλινδρο
- Μπορούμε να "κρύψουμε" μέρος του χρόνου ανάγνωσης από το δίσκο χρησιμοποιώντας λίγο περισσότερη μνήμη
  - Double buffering

Double Buffering

- Problem: Have a File
  - Sequence of Blocks B1, B2

- Have a Program
  - Process B1
  - Process B2
  - Process B3

  (σειριακή ανάγνωση)

Single Buffer Solution

(1) Read B1 $\rightarrow$ Buffer

(2) Process Data in Buffer

(3) Read B2 $\rightarrow$ Buffer

(4) Process Data in Buffer ...

Single Buffer Time

Έστω: P = χρόνος επεξεργασίας/σελίδα

R = χρόνος για να διαβάσω 1 σελίδα (υποθέτω τυχαία προσπέλαση)

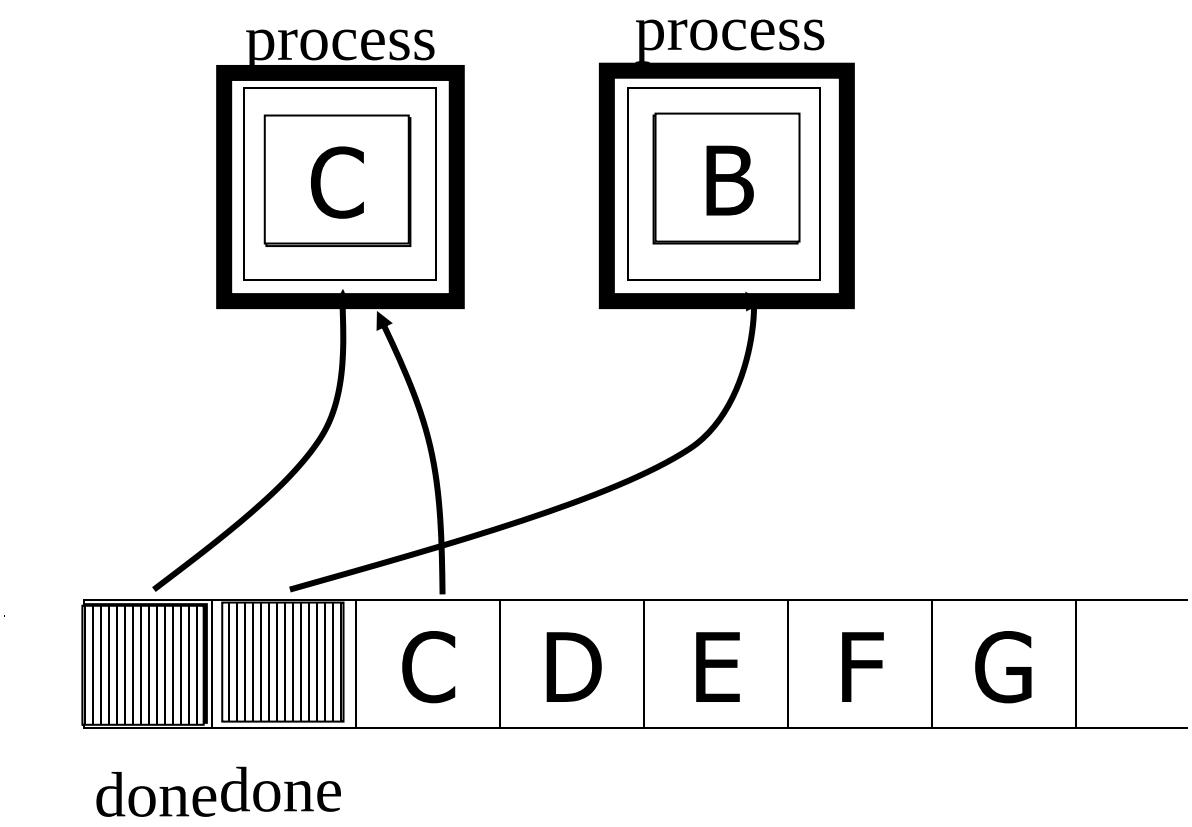
n = # αριθμός σελίδων

Single buffer time = n(P+R)

Double Buffering

Memory:

Disk:



Double Buffering Time

P = χρόνος επεξεργασίας/σελίδα
R = χρόνος για να διαβάσω 1 σελίδα
(υποθέτω τυχαία προσπέλαση)
n = αριθμός σελίδων

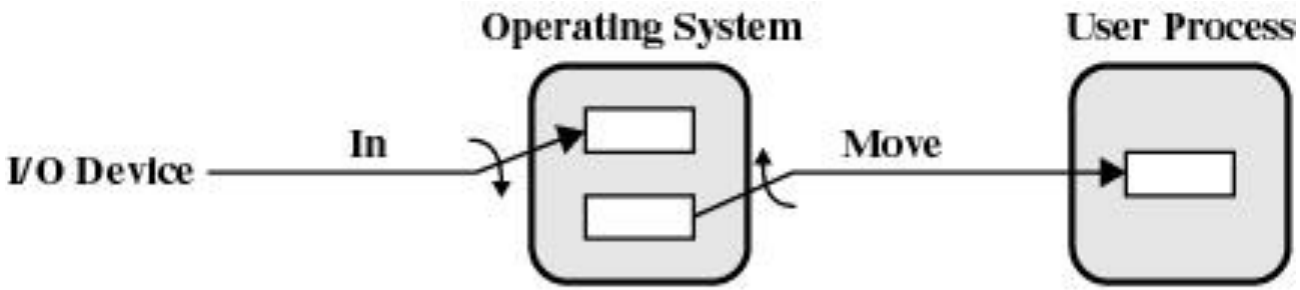
Υποθέτω $P \geq R$

- Double buffering time = R + nP
- Single buffering time = n(R+P)

More on Prefetching (application level)

- Requires efficient asynchronous I/O implementation
- Data dependant scheduling of I/O
- DB process queues up several requests
  - Better chance for disk controller to optimize seeks
  - Better utilization of multiple disks (allow them to work in parallel)
- Big savings in CPU intensive tasks (e.g. *sorting*)

Circular Buffering



(c) Double buffering

(d) Circular buffering

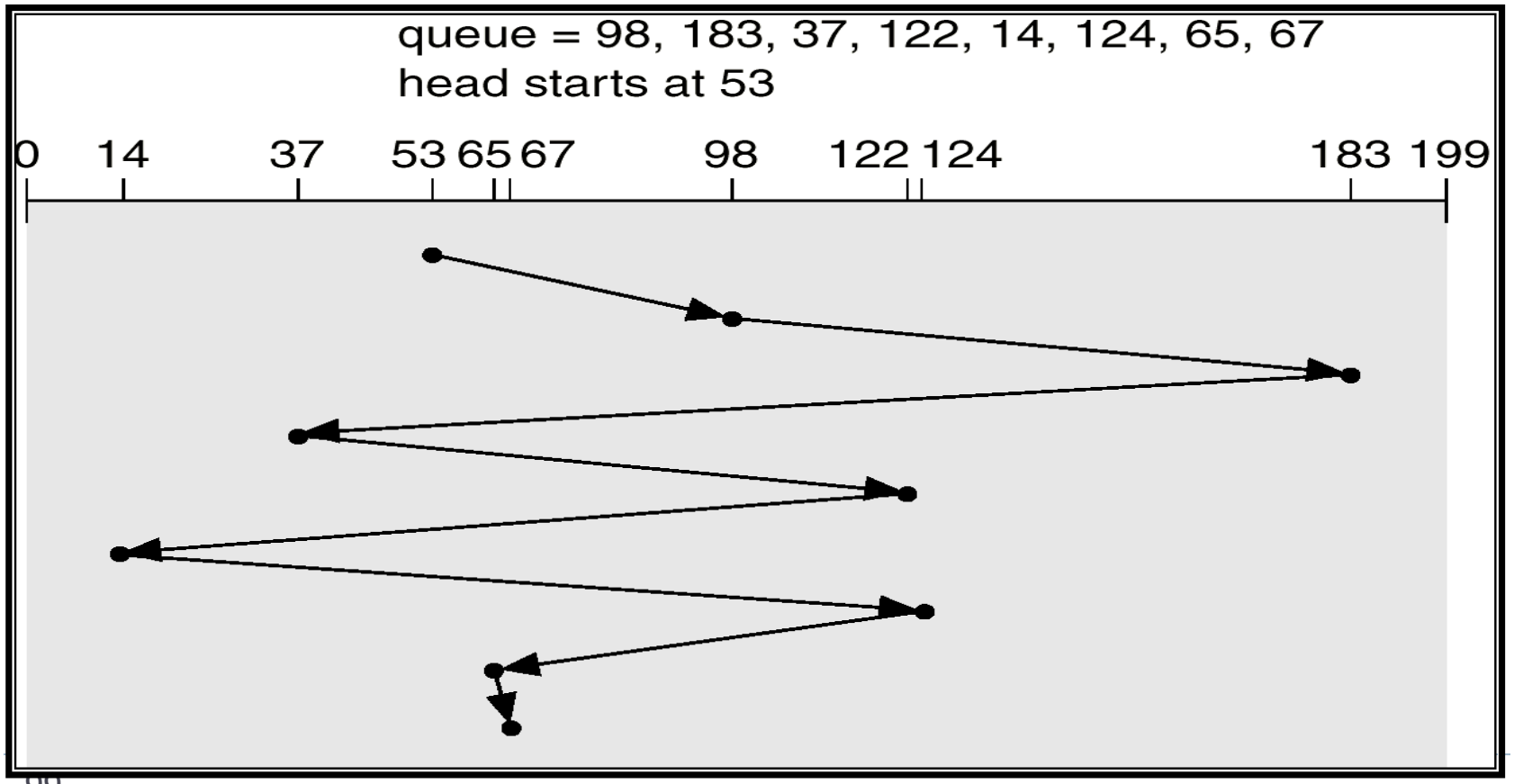
I/O Buffering Schemes (input)

Περισσότερες βελτιστοποιήσεις για καλύτερη απόδοση του υποσυστήματος δίσκων

- Disk Scheduling Algorithm
  - Elevator algorithm
- Disk Arrays (RAID)
- On Disk Cache

First- Come-First-Served (FCFS)

- Δίκαιη πολιτική
- Αργή

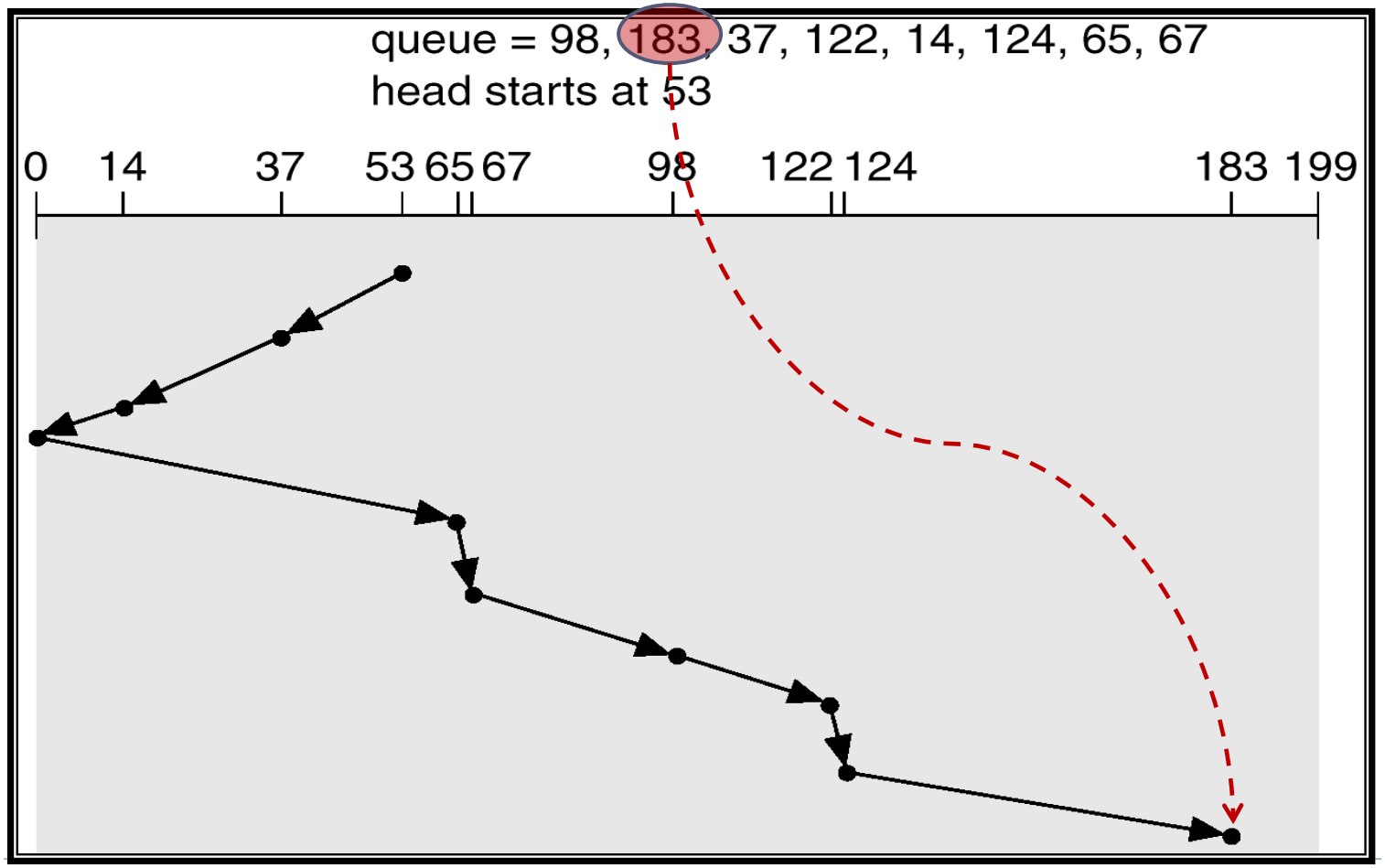


SCAN or Elevator Algorithm

- Η κεφαλή του δίσκου ξεκινάει από τη μία άκρη (πχ έξω) και κινείται προς την άλλη (πχ μέσα) εξυπηρετώντας αιτήσεις. Μόλις φτάσει στο άλλο άκρο αλλάζει κατεύθυνση
  - Συνολικά καλύτερη απόδοση (μειώνει το seek time)
  - Κάποιες αιτήσεις για σελίδες ενδέχεται να περιμένουν αρκετά μέχρι η κεφαλή να αλλάξει κατεύθυνση
  - Νεώτερες αιτήσεις μπορεί να εξυπηρετηθούν νωρίτερα από άλλες που περιμένουν περισσότερη ώρα

SCAN

•Total head movement=208 cylinders



Solid State Disks (SSD)

- Χρήση μνήμης flash, απουσία κινούμενων μερών
  - Σημαντικά ταχύτερη τυχαία προσπέλαση (random I/O) σε σύγκριση με HDD
  - Μικρότερες απαιτήσεις για ψύξη (όχι πάντα)
  - Καθόλου θόρυβος
- Αρκετά ακριβότεροι από HDD (€/GB)
  - Κατανάλωση ρεύματός μπορεί να είναι μικρότερη ή μεγαλύτερη από ένα HDD (ανάλογα με τη χρήση)

Solid State Disks (SSD)

- Υποστηρίζουν συγκεκριμένο αριθμό εγγραφών-επανεγγραφών (wearing)
  - Μπορώ να γράψω άμεσα σε μία κενή σελίδα (4KB) αλλά αν αυτή δεν είναι κενή η επανεγγραφή γίνεται σε επίπεδο πολλών σελίδων (πχ 128) κάνοντας τες πρώτα erase ακόμα και αν αλλάζει μία μόνο σελίδα 4K
  - Υπάρχει ένας (σχετικά μεγάλος) αριθμός από erase cycles μετά τα οποία δε μπορώ να χρησιμοποιήσω ξανά το block
    - Στην πράξη ο αριθμός αυτός μπορεί να αναλογεί σε μερικά PB από writes (τα οποία ποτέ δε θα δω σε desktop χρήση)
    - Σε ένα data center όμως?
- Σταδιακή μείωση της απόδοσης, ανάγκη για αναδιάταξη (Garbage Collection, TRIM)

Παράδειγμα Ομάδα 16 σελίδων των 4KB

Ομάδα Χ

FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE

Γράψε σελίδες A,B,C,D

Ομάδα Χ

A	B	C	D
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE

Παράδειγμα: ομάδα 16 σελίδων των 4KB

Γράψε σελίδες E,F,G,H

A	B	C	D
E	F	G	H
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE

Τροποποίησε A',B',C',D'

A	B	C	D
E	F	G	H
A'	B'	C'	D'
FREE	FREE	FREE	FREE

Mark as Stale

Για να γράψω πάλι εδώ πρέπει να κάνω erase όλη την ομάδα

Garbage Collection: Πως ξανακερδίσω τα stale pages?

Ομάδα X

A	B	C	D
E	F	G	H
A’	B’	C’	D’
FREE	FREE	FREE	FREE

Ομάδα Y

FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE

Κάνε erase όλη τη X

FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE

Αντέγραψε στην Y τα χρήσιμα pages της X

E	F	G	H
A’	B’	C’	D’
FREE	FREE	FREE	FREE
FREE	FREE	FREE	FREE