Advanced Database Management Systems

Lecture 13 –13.1-13.3 & Appendix B
Physical Storage

Cost Analysis

- What is the cost of the following problems?

1. Given 100,000 unordered names count all the names beginning with ‘R’.
2. Given 100,000 names (unordered) find the alphabetic median.
3. Given 10,000,000 insurance customer records, find a customer with the median premium.
4. Given 250,000,000 US tax records, find the citizen with the largest tax payment in 2007.
5. Given 250,000,000 US tax records, find a citizen with the median tax payment in 2007.

Cost Analysis

		data in memory	moving data from disk to memory
1	O(n) n=100,000	100,000 comparisons	100 disk block reads
2	O(n log(n)) n=100,000	1,000,000 comparisons	100 disk block reads
3	O(n log(n)) n=10,000,000	100,000,000 comparisons	10,000 disk block reads
4	O(n) n=250,000,000	250,000,000 comparisons	250,000 disk block reads
5	O(n log(n)) n=250,000,000	2,500,000,000 comparisons	250,000 disk block reads

Rough estimate: 1000 records / disk block

Which takes more time, the in memory algorithm, or the disk reads?

Cost Analysis

		data in memory	moving data from disk to memory?
1	O(n) n=100,000	100,000 comparisons .0001 sec	100 disk reads .1 sec
2	O(n log(n)) n=100,000	1,000,000 comparisons .001 sec	100 disk reads .1 sec
3	O(n log(n)) n=10,000,000	160,000,000 comparisons .1 sec	10,000 disk reads 10 sec
4	O(n) n=250,000,000	250,000,000 comparisons .25 sec	250,000 disk reads 4 min
5	O(n log(n)) n=250,000,000	4,800,000,000 comparisons 4.8 sec	250,000 disk reads 4 min (but will it all fit in memory?)

Assume 1 GHz (10^9 ops/sec) processor
Assume 1 msec block (10^3 blocks/sec) transfer time

Joins are Expensive

- R: 300,000,000 US citizens $3x10^8$
- T: 8,000,000 airline passengers $8x10^6$
- Join on name, count flights per citizen

- Naïve solution: cross product → $2x10^{15}$ records
  - No way it will fit in memory

- Better solution:
  - Read each citizen once,
    then read each passenger 300,000,000 times

Inefficient Join Algorithm

```
for c = 0, citizens.size()
    citizen_record = read_from_disk(citizens, c)
    flights = 0
    for p = 0, passengers.size()
        passenger_record = read_from_disk(passengers, p)
        if (citizen_record.name == passenger_record.name)
            flights++;
    write_to_disk(flight_count, citizen_record.name, flights)
```

10^{15} reads → 10^{12} seconds → 10^{8} hours

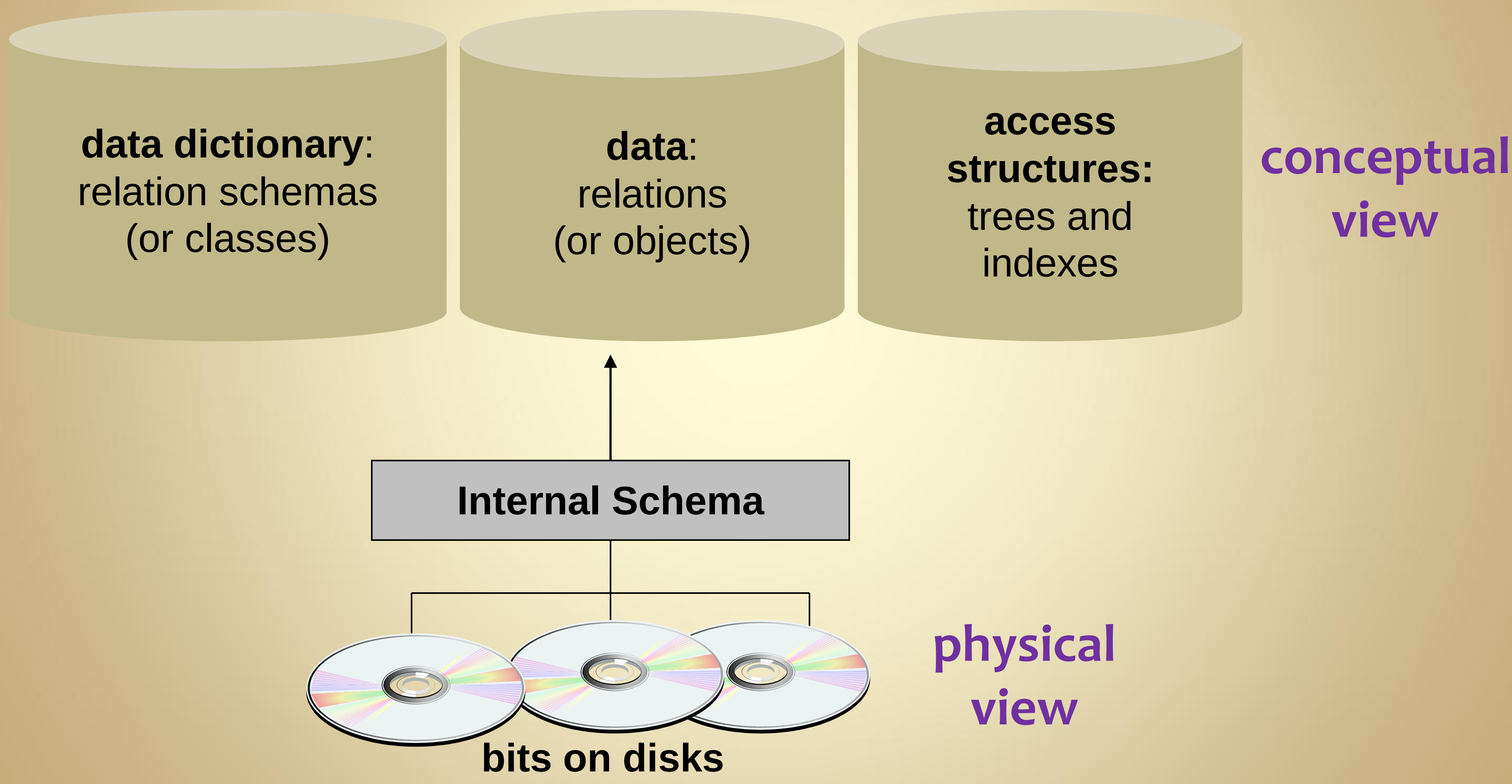
Clearly, we need a better algorithm

Physical Data Independence

- conceptual schema (tables)
  and external schema (views)
  are not affected by changes
  to the physical layout of the data

- Database (application) designers still need to understand the internals of the DBMS
  - to optimize performance
  - to perform maintenance
  - data structures and algorithms are applicable in other areas of computer science

Storage: Bits and Bytes



Storage Media

- electronic storage (cache, main memory)
  volatile fast (speed of light)
- flash memory (USB drives)
  non-volatile fast (limited by USB)
- magnetic disks
  non-volatile slow (moving parts)
- optical disks (CD-ROM, DVD)
  non-volatile slow (moving parts)
- tape
  non-volatile very slow (moving parts, sequential access)

Storage Media Prices

	typical retail price	GB / $1
main memory	1GB / $50	0.02
flash memory	32GB / $65	0.5
magnetic disk	500GB / $60	8
CD	100*700MB / $20	7
DVD	100*5GB / $35	10
tape	100GB / $20	10

exponent prefixes

tera	10^{12}
giga	10^{9}
mega	10^{6}
kilo	10^{3}
milli	10^{-3}
micro	10^{-6}
nano	10^{-9}
pico	10^{-12}

Database Storage Needs

- To support a DBMS, the data store must be
  - non-volatile
  - readable and writeable
  - random access
  - cheap (large amounts required)

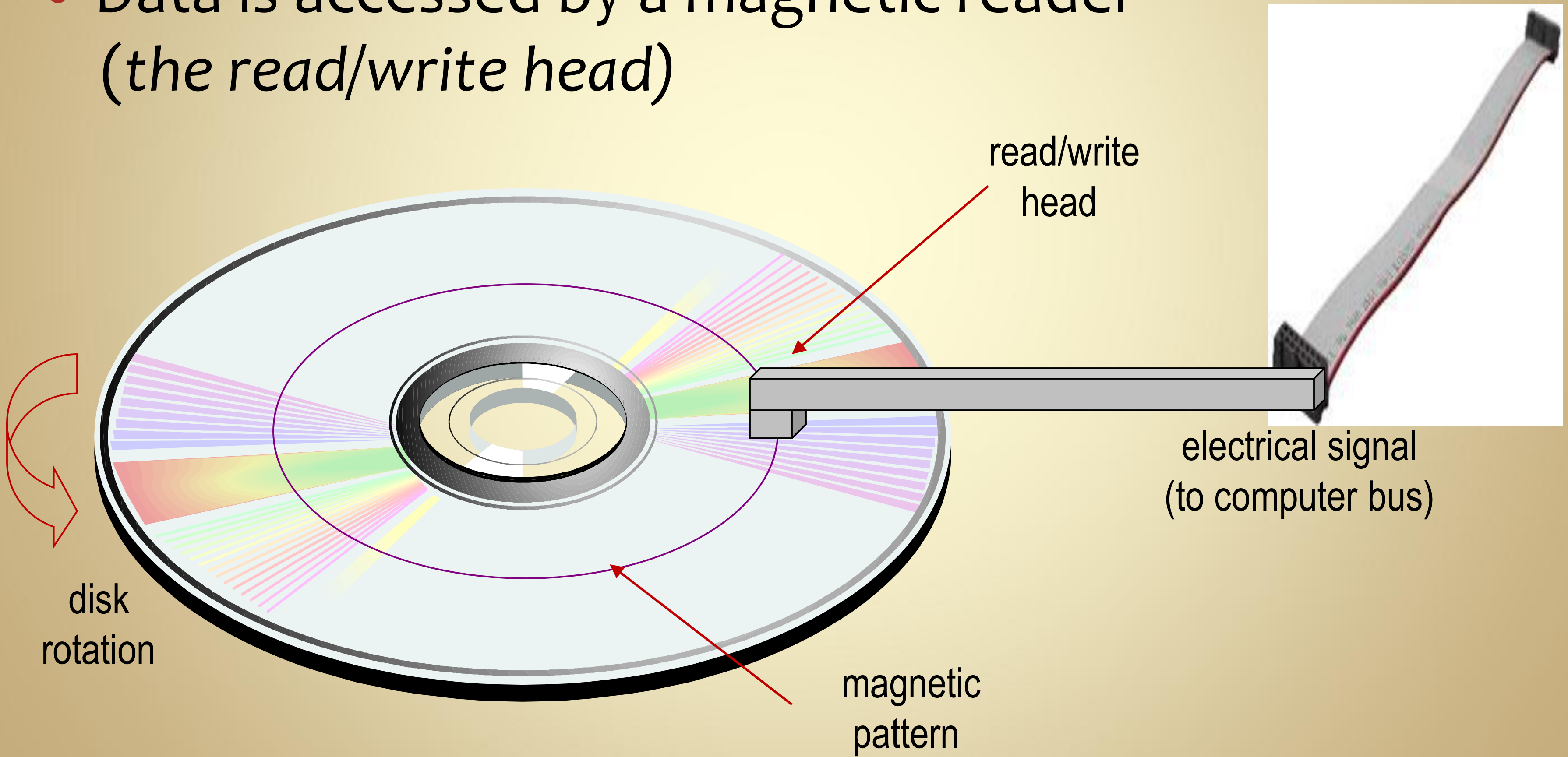
- A DBMS needs magnetic disk storage

  - consequence: internal schema must be designed
    to optimize access in order
    to minimize the effect of slow physical parts

  - corollary: We need to know how a disk works:
    parameters that define access time
    are needed to optimize performance

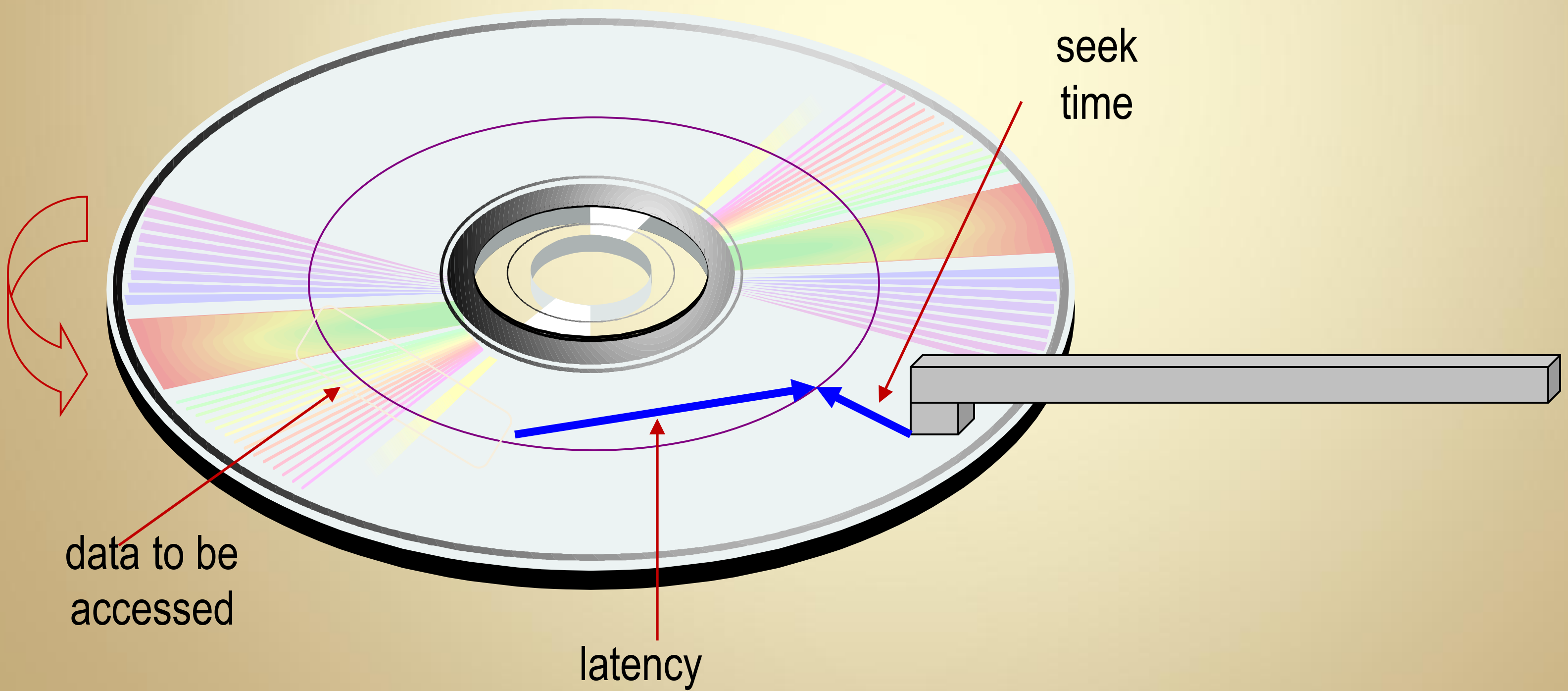
Disk Mechanics

- Data is stored in concentric *tracks*
- Data is accessed by a magnetic reader (*the read/write head*)



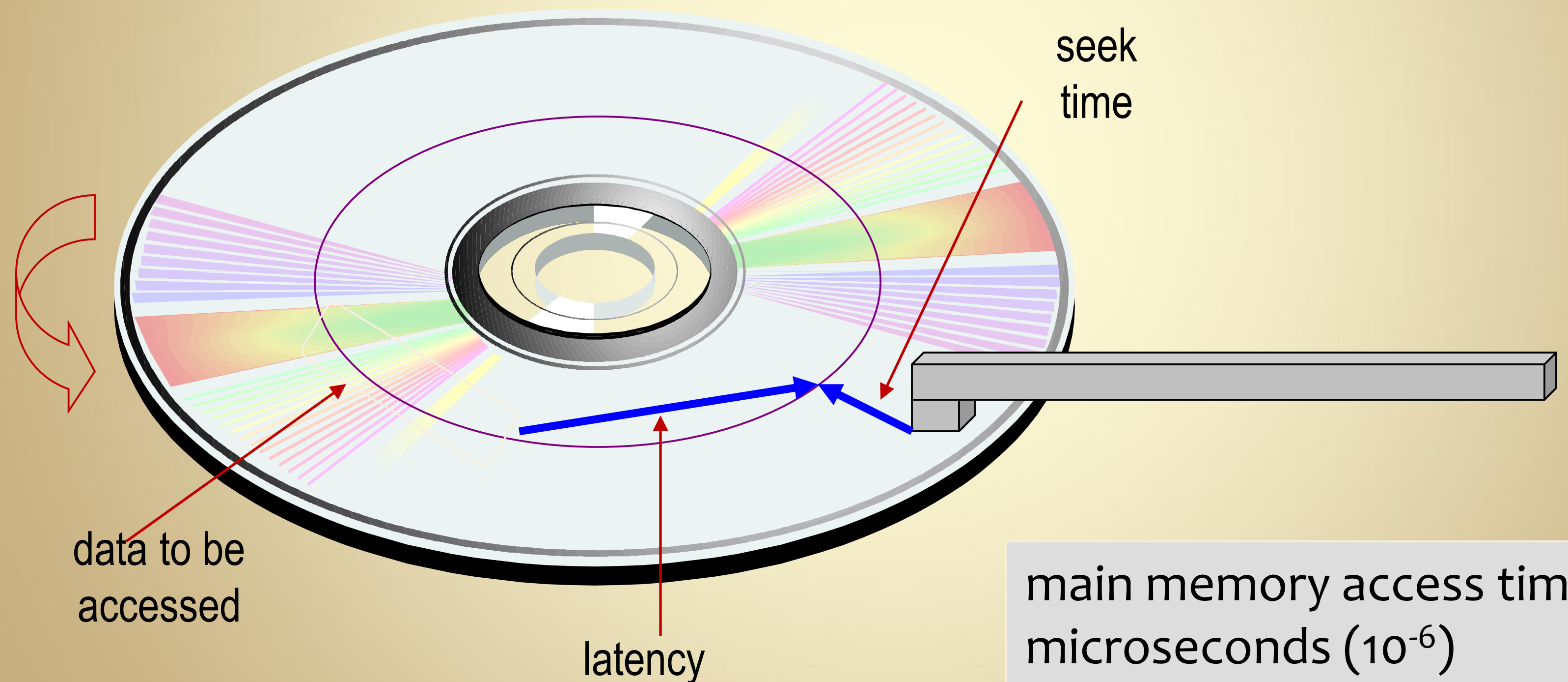
Disk Data Access

- To read or write data on a disk, the head must
  - move to the correct track (seek time)
  - wait for rotation to move data to it (latency)



Seek Time and Latency

- Average seek time: s = 3-4 msec
- Average Latency (rotational delay): rd = 2-3 msec



main memory access time is microseconds (10^{-6}) or nanoseconds (10^{-9})

Disk Speed and Rot. Delay

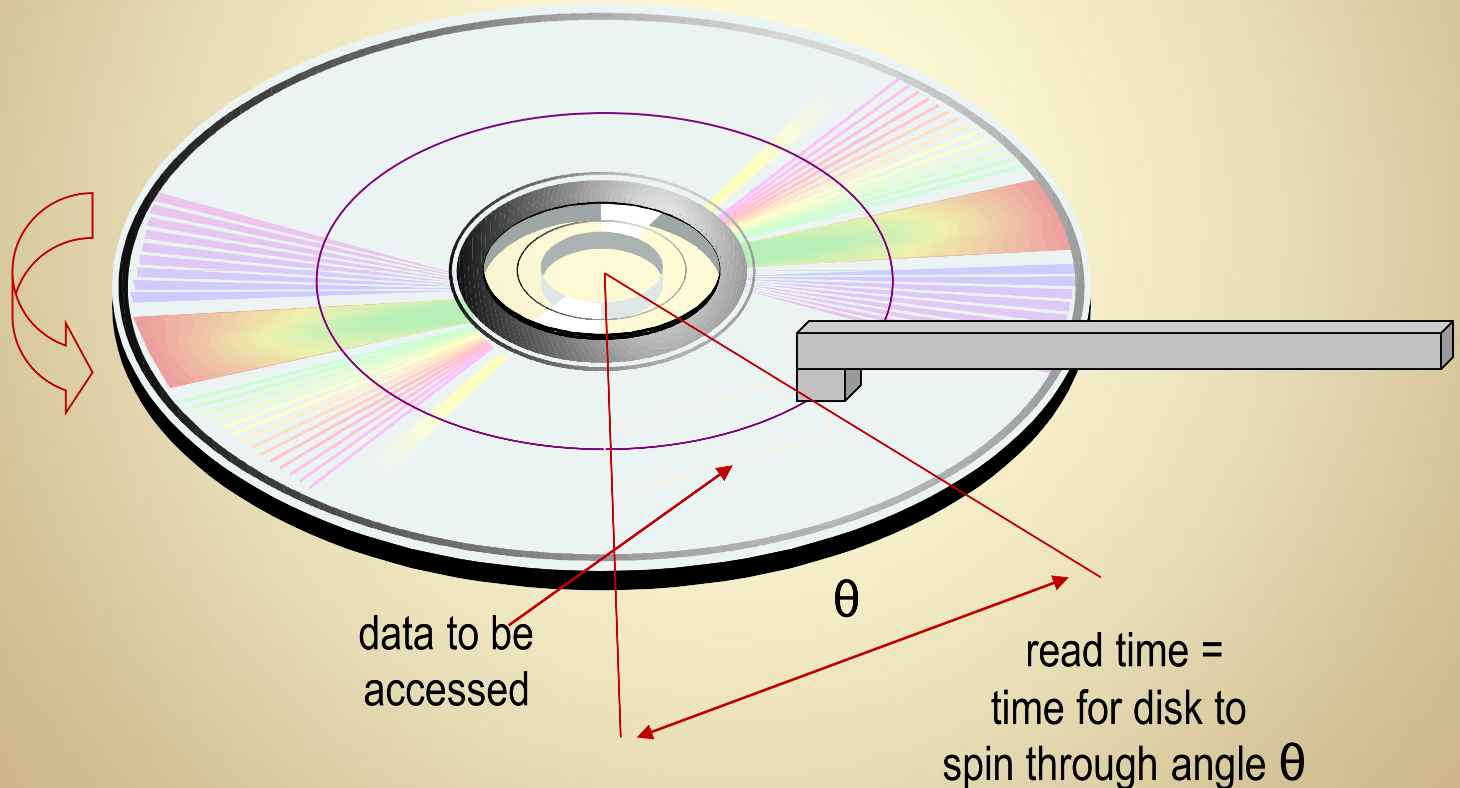
- Disk speed (p) is typically given in RPM
  - p = 10,000 rotations/minute is typical
- Rotational delay (rd) is the time for ½ rotation
  - rd = 0.5 / p

$$\frac{0.5}{10000rpm} = \frac{0.5\,\mathrm{min}}{10000} \times \frac{60\,\mathrm{sec}}{\mathrm{min}} = 0.003\mathrm{sec} = 3\mathrm{msec}$$

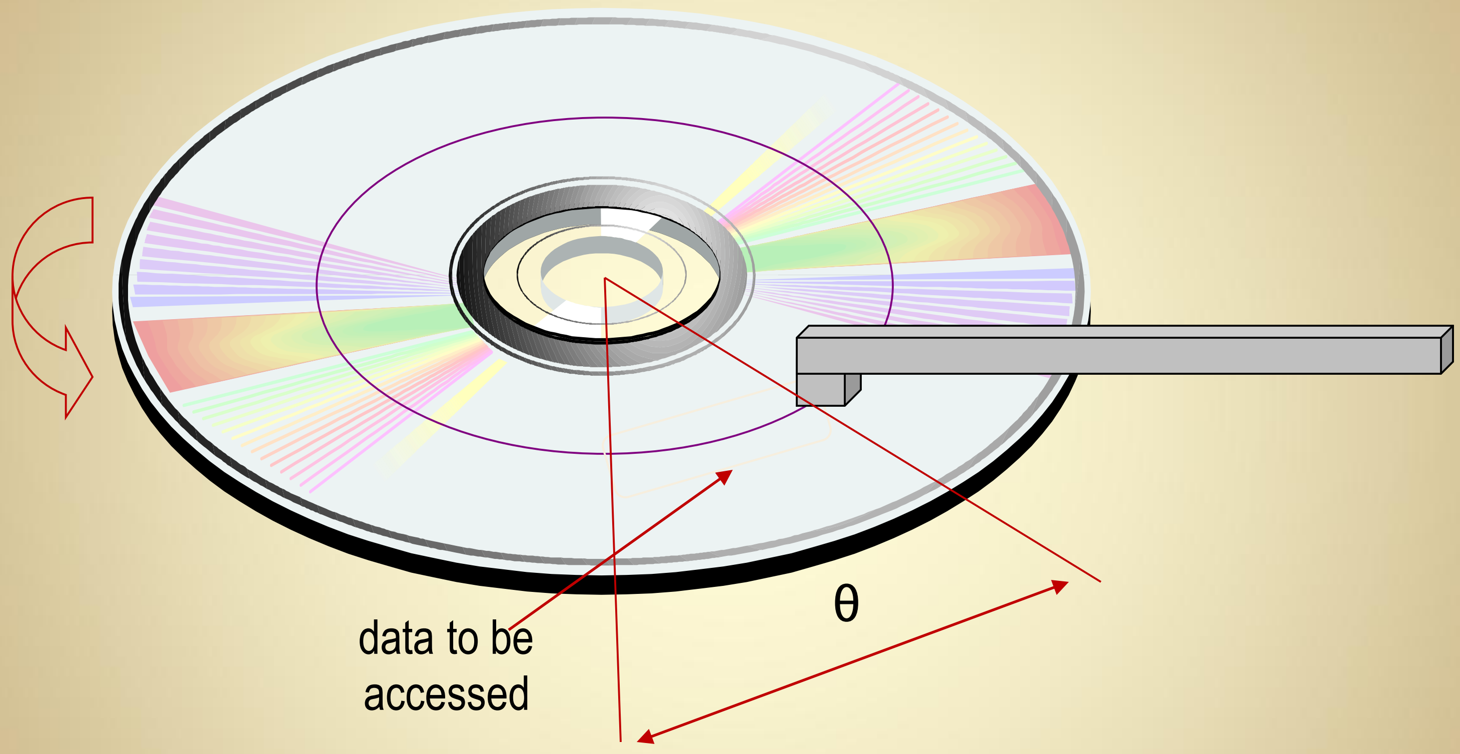
- Time for one full rotation
  is twice the rotational delay (2 * rd)

Reading Data from Disk

- Once start of data is located, it must be read or written



Angular Spin Time



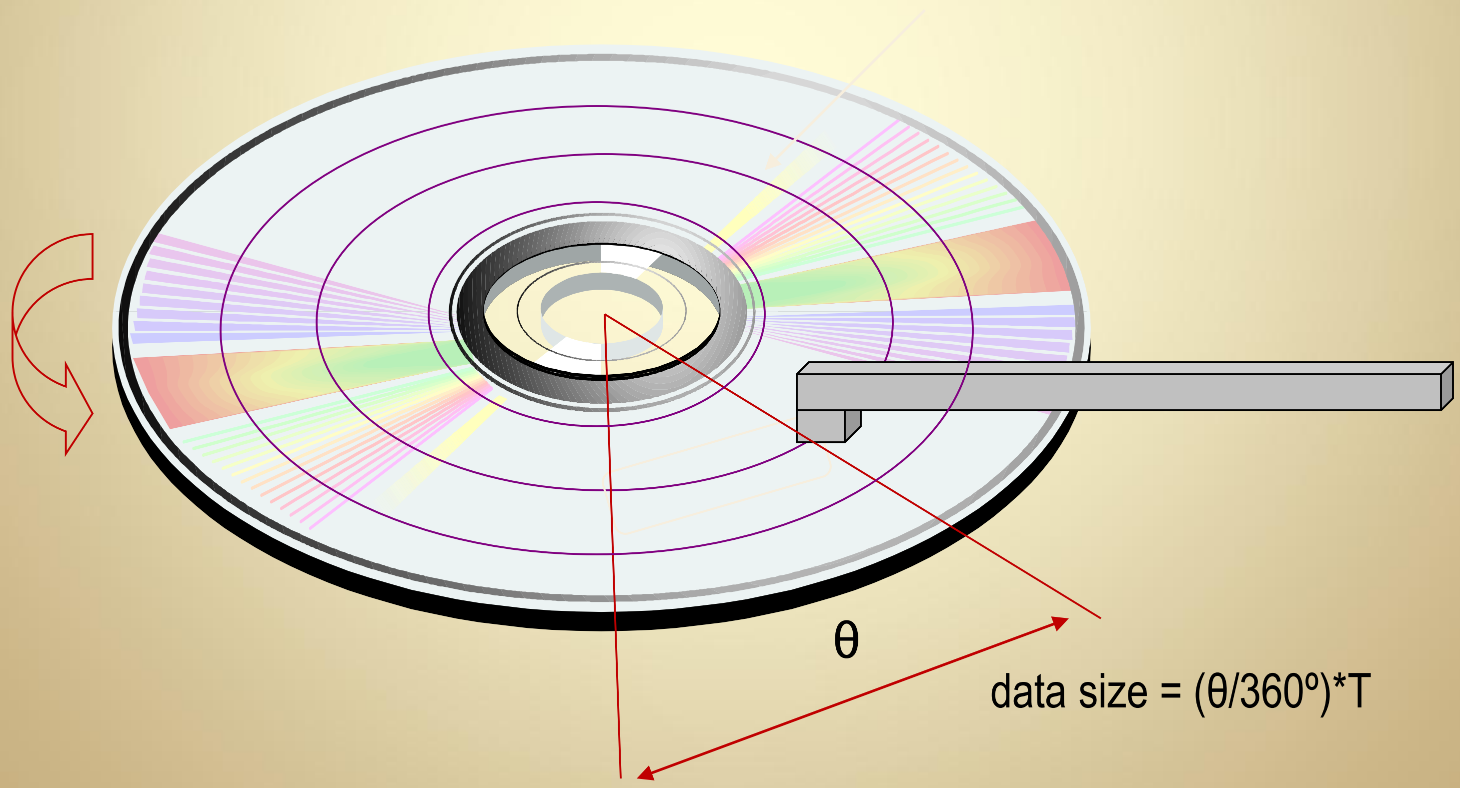
data transfer time: (θ/360º) * (2*rd) = (θ/360º) / p

Track Size

Assumption: all tracks hold the same amount of data and disk velocity is constant → constant transfer rate.

T = track size

example = 50,000 bytes/track

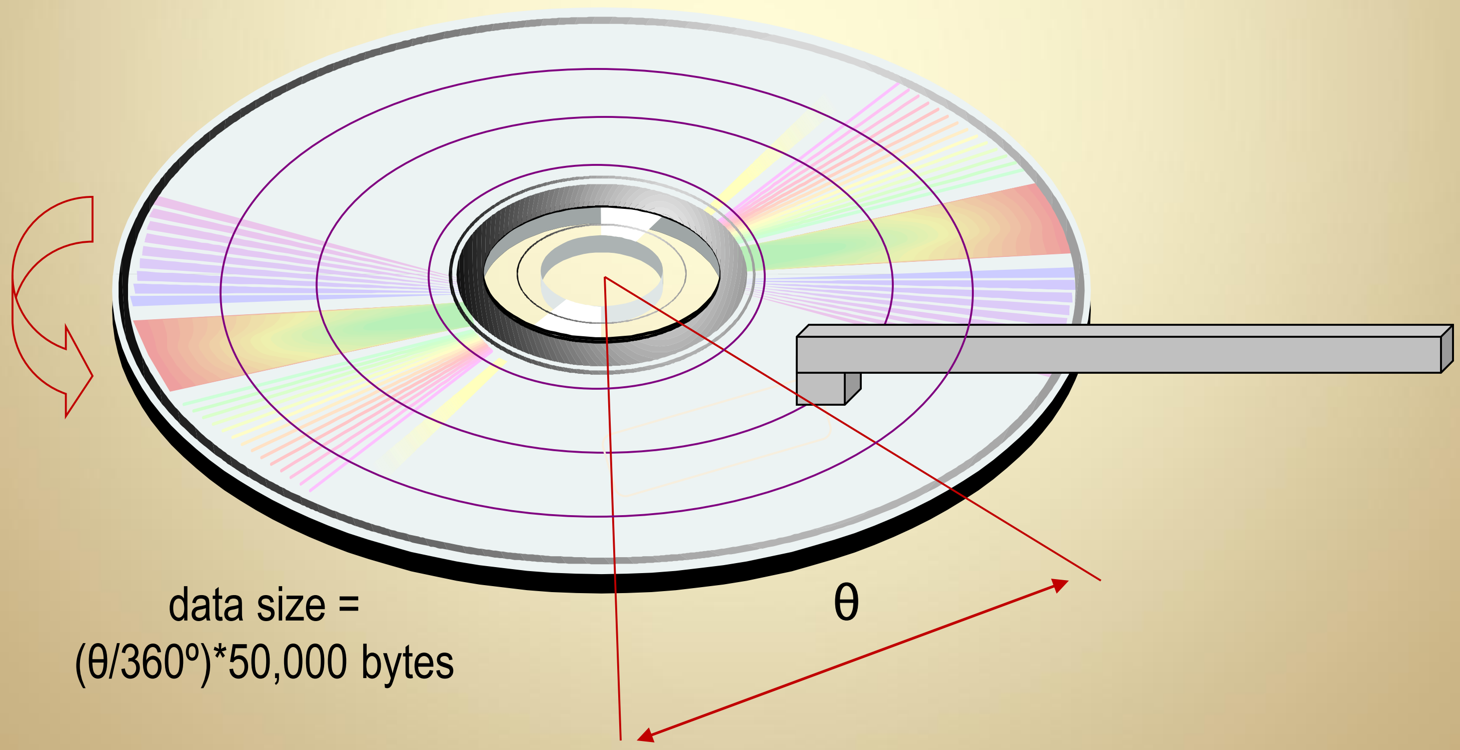


data size = (θ/360º)*T

Data Transfer Rate

track size: T = 50,000 bytes/track
velocity: p = 10000 rpm = 167 tracks/sec

transfer rate: tr = T * p = T /(2*rd)



Data Transfer Rate

track size: T = 50,000 bytes/track
velocity: p = 10000 rpm = 167 tracks/sec

transfer rate (tr):

$$\begin{aligned} tr &= T*p \\ &= 50000 \text{ bytes/track} * 167 \text{ tracks/sec} \\ &= 8{,}350{,}000 \text{ bytes/sec} = 8 \text{ MB/sec} \end{aligned}$$

alternate calculation:

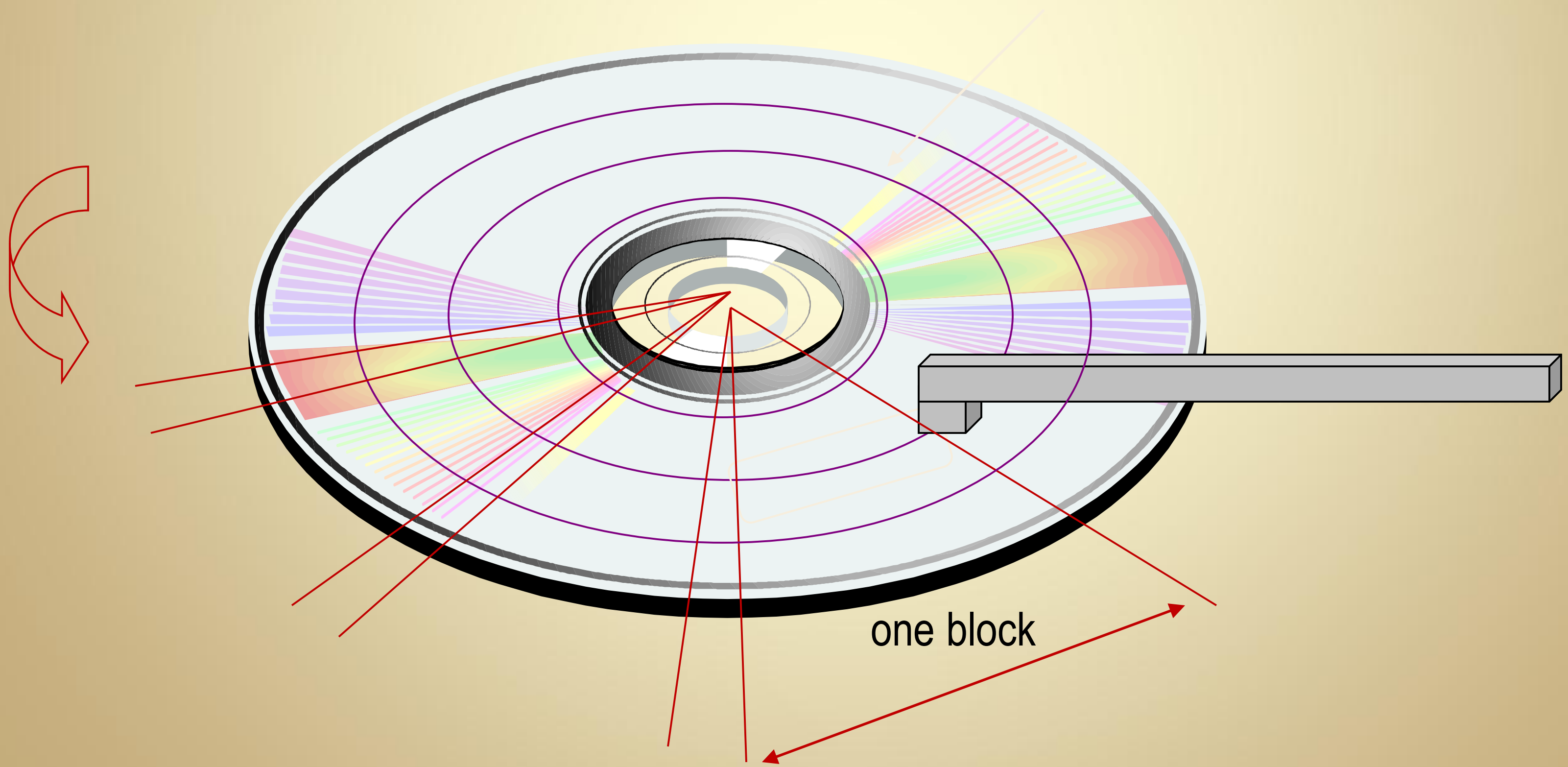
$$\begin{aligned} tr &= T/(2*rd) \\ &= 50000/(2*3e\text{-}3) \\ &= 8 \text{ MB/sec} \end{aligned}$$

Blocking

Tracks are divided into *blocks*, separated by *inter-block gaps*

typical block size: B = 1024 bytes (50 blocks/track)

typical gap size: G = 128 bytes

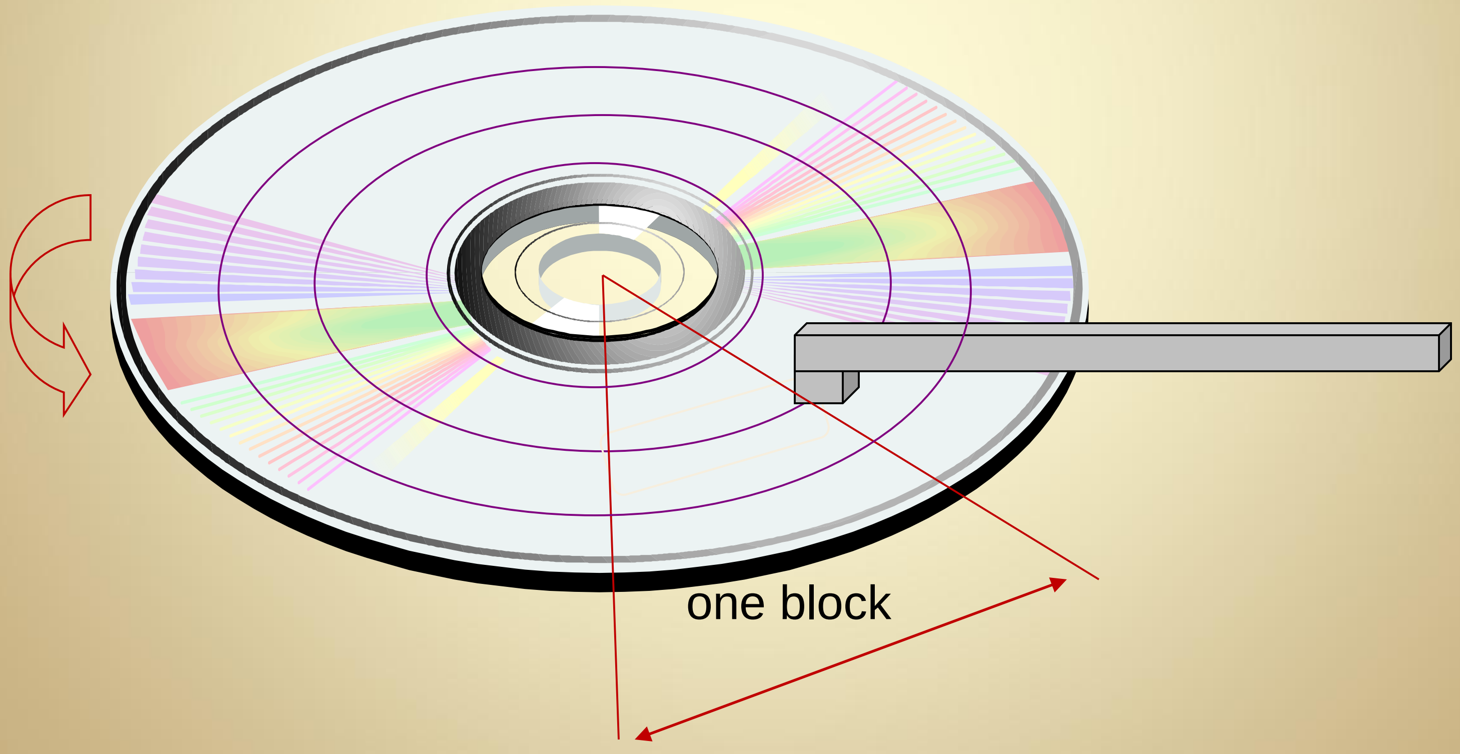


Block Transfer Rate

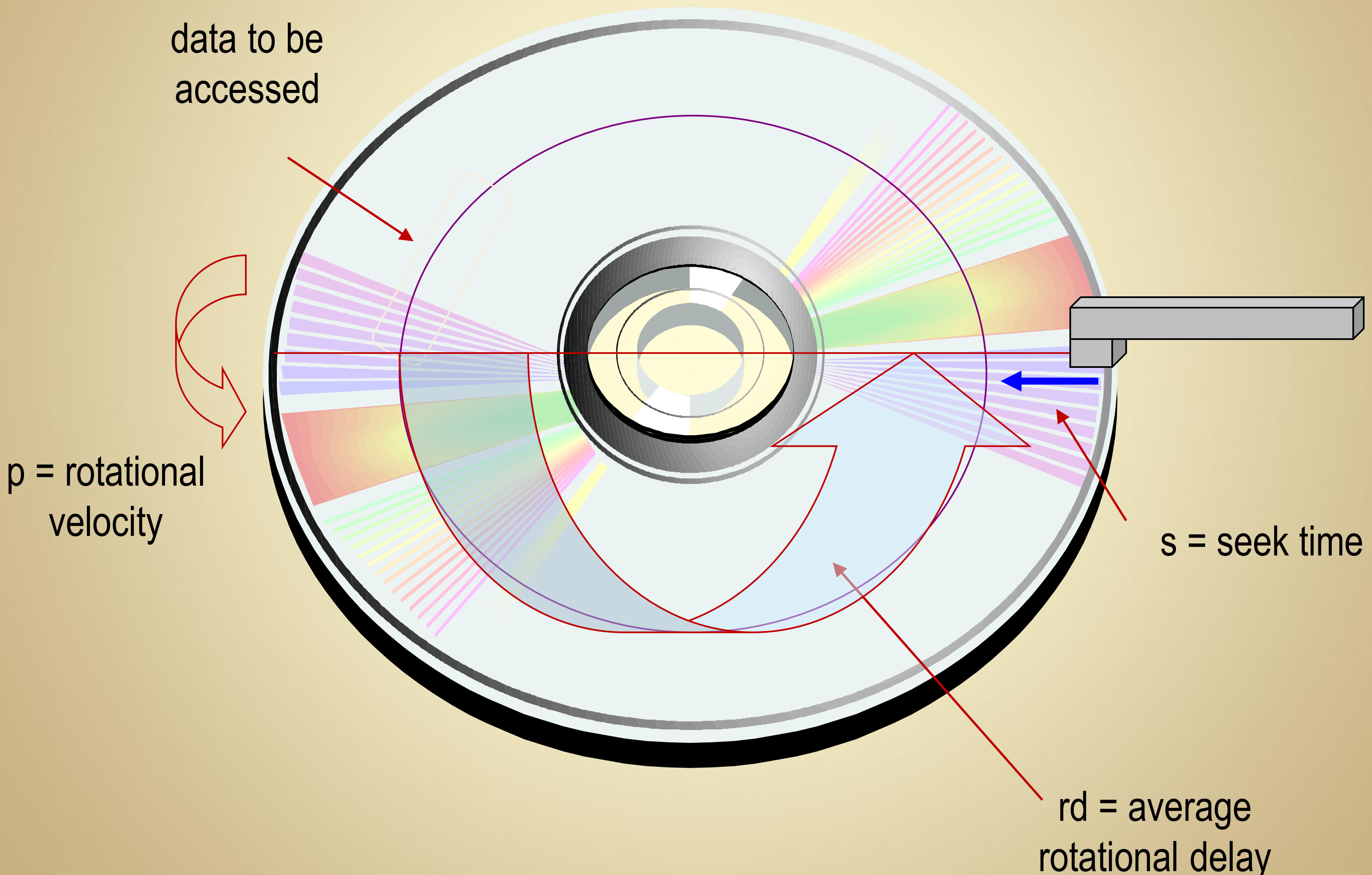
block transfer time:

btt = B / tr =

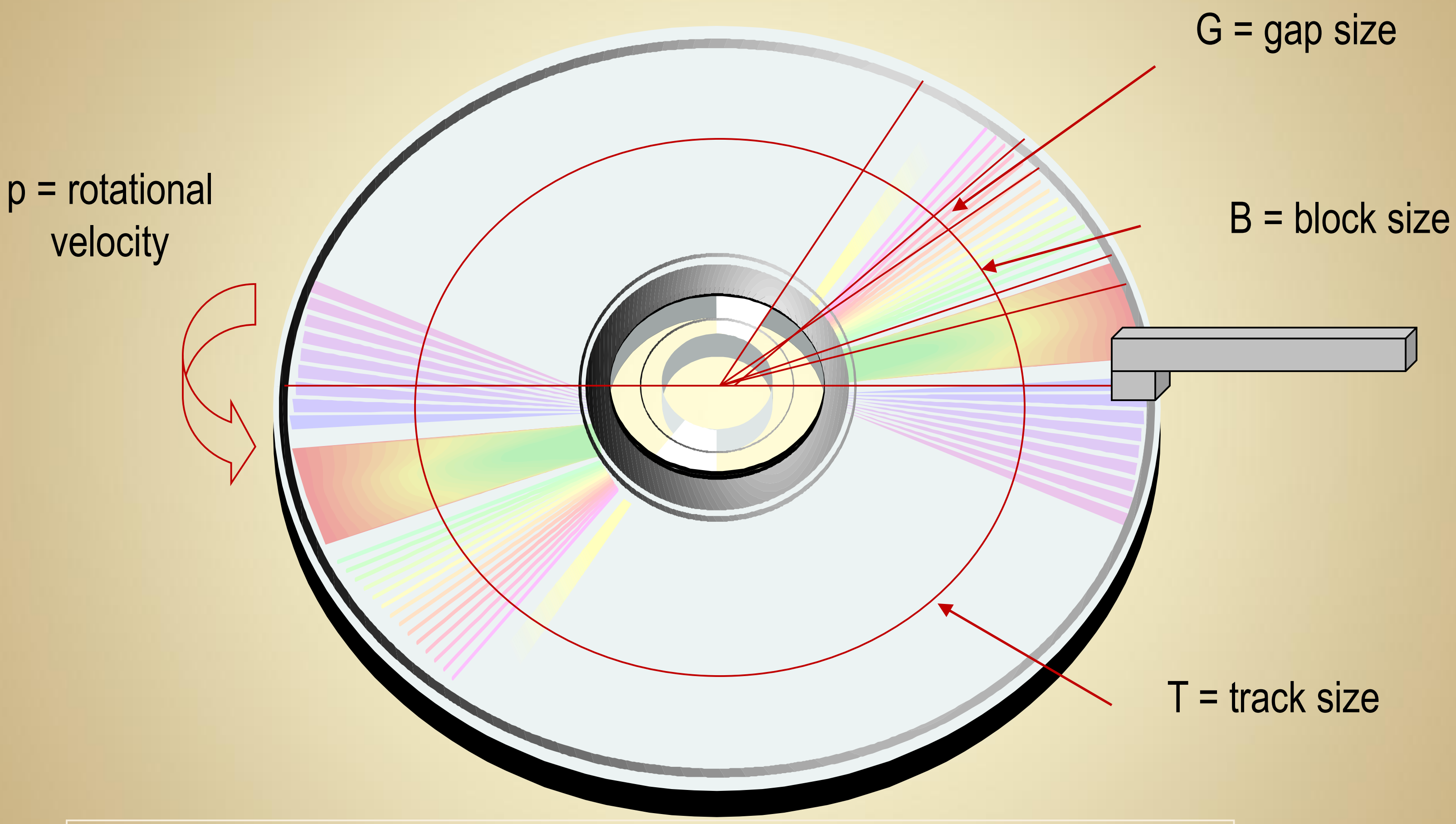
= 1024 bytes / (8Mb/sec) = 0.128 msec



Seek Time, Latency and Rotational Velocity



Block Transfer Time

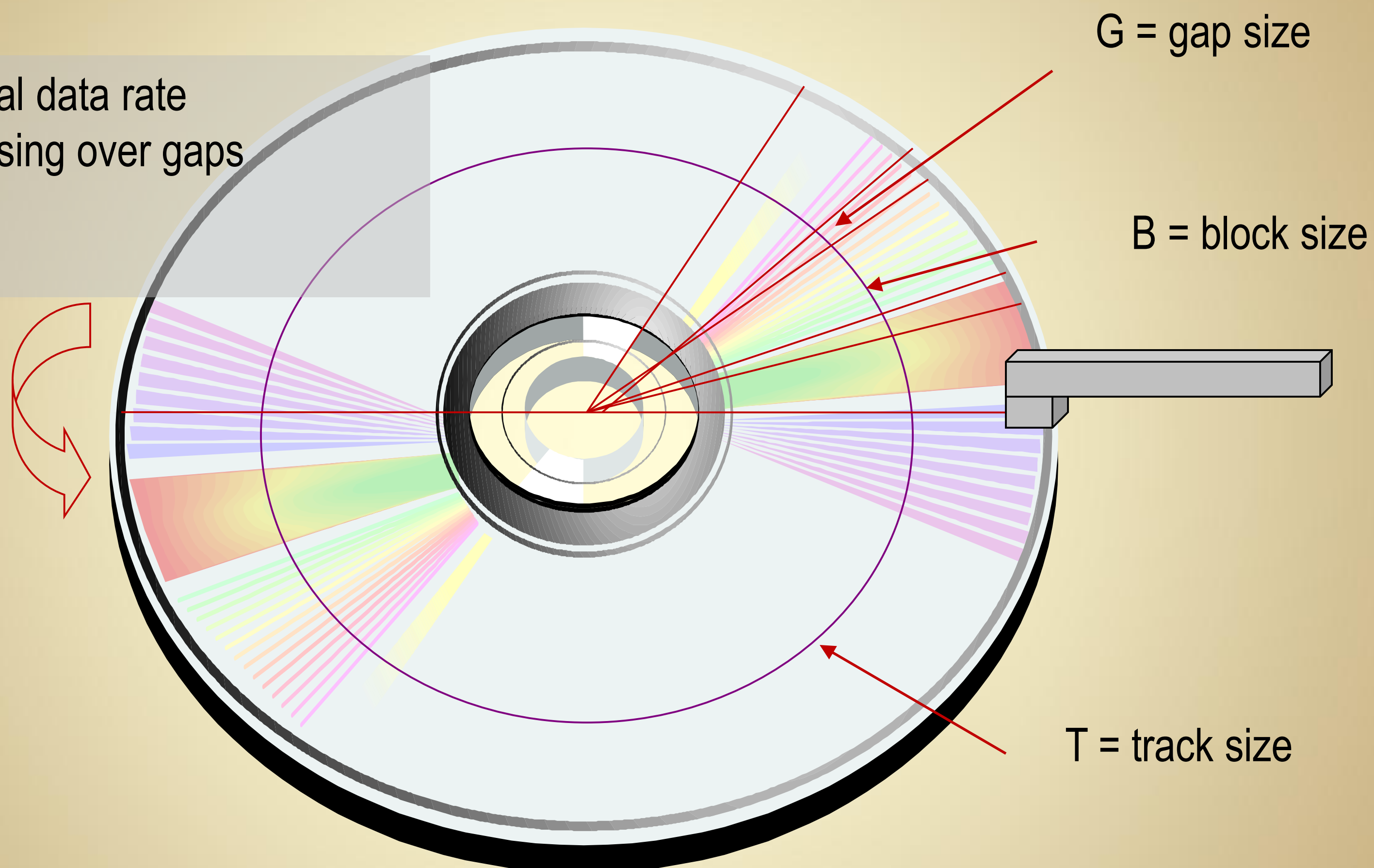


btt = time for one block to pass under read/write head

= B/tr

Bulk Transfer Rate

btr = best actual data rate since time passing over gaps is "wasted"



btr = transfer rate for *consecutive* blocks = $\dfrac{B}{B+G} \times tr$

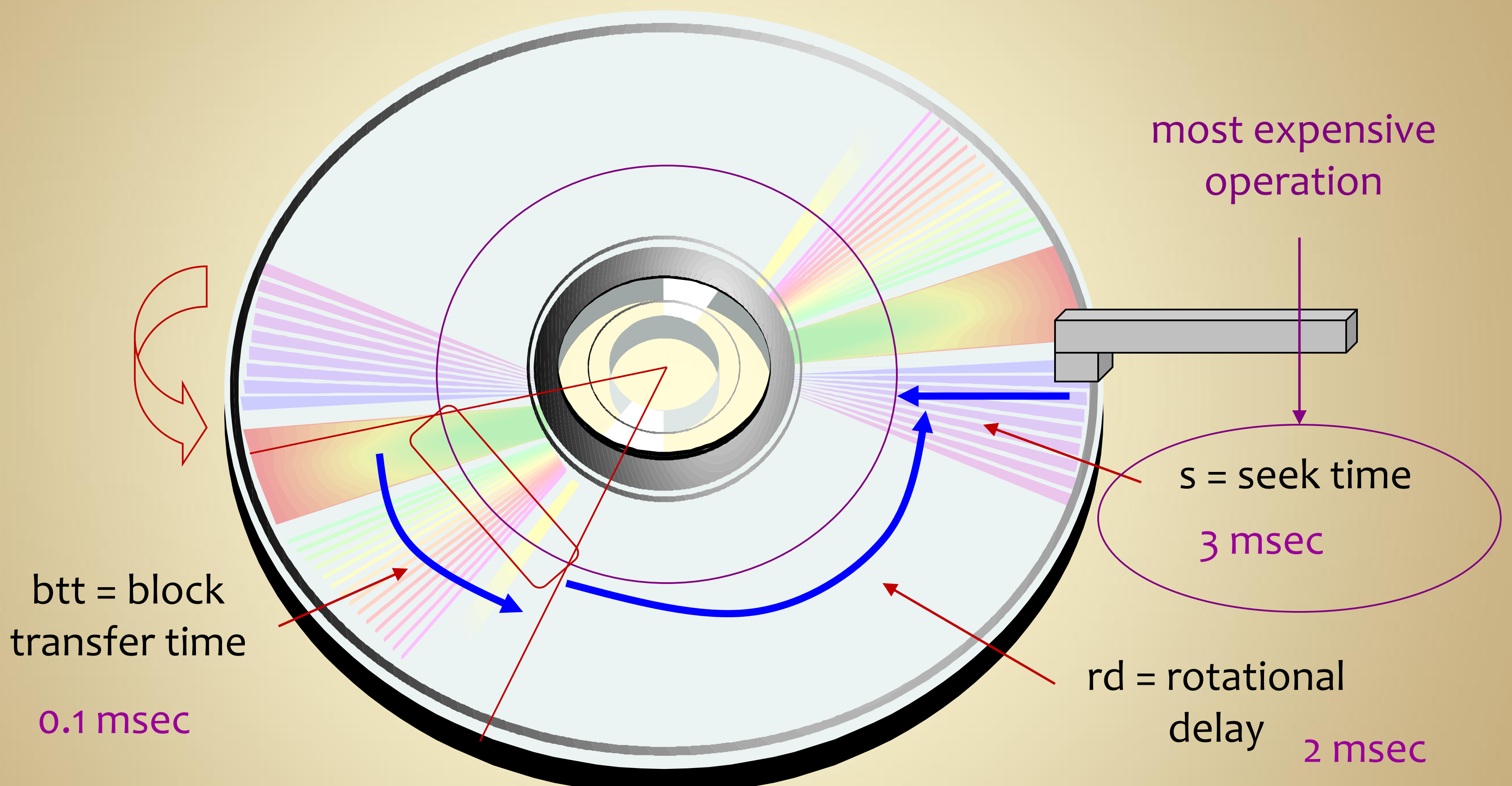
Bulk Transfer Rate

- btr = best actual data rate,
  since time passing over gaps is "wasted"

- For our purposes, we'll generally
  ignore the gaps to simplify the computations, thus

btr = transfer rate for *consecutive* blocks = B * tr

Random Block Transfer Time



rbtt = time to locate and transfer one random block
= s + rd + btt (5.1msec)

Transferring Multiple Blocks

Time to transfer n blocks of data:

if blocks are randomly located, we pay seek and latency for each block:

rbtt = n*(s + rd + btt)

if blocks are consecutively located, we only pay seek and latency once

cbtt = s + rd + n*btt

Fundamental Results

- Organize data in blocks
  - this is the basic unit of transfer
- Whenever possible, layout data to maximize possibility of consecutive block retrieval
  - avoids seek and latency costs
- This will impact
  - record layout in files
  - access structure (indexes, trees) organization

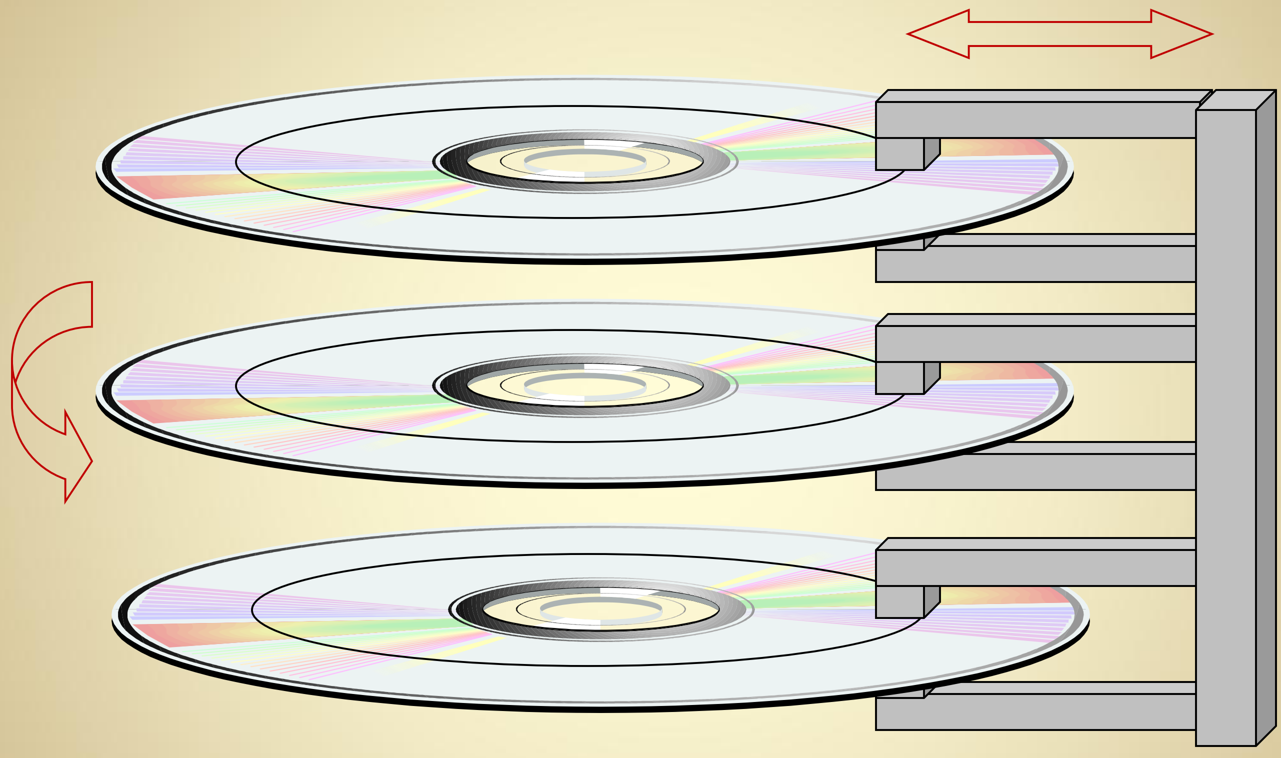
Disk Parameters

parameter		typical value	source
s	seek time	3 msec	fixed
p	rotational velocity	10,000 rpm 167 rps	fixed
rd	rotational delay (latency)	2 msec	.5*(1/p) (average)
T	track size	50 Kbytes	fixed
B	block size	512-4096 bytes	formatted
G	interblock gap size	128 bytes	formatted
tr	transfer rate	800Kbytes/sec	T*p
btt	block transfer time	1 msec	B/tr
btr	bulk transfer rate (consecutive blocks)	700Kbytes/sec	(B/(B+G))*tr

Disk Packs

- Typical disk drives have multiple disk surfaces
  - surfaces are sometimes called platters
- Disks are connected to same spindle
  - disks rotate together
- Each surface has its own read/write head
  - Heads are connected to single motor,
    they all move together
- We can read/write the same block
  on multiple disks simultaneously

Cylinders



a *cylinder* is made up of the same track on all platters