4. Accessing the Data

In this chapter we go into some details:

- deep into the (runtime) system
- close to the hardware

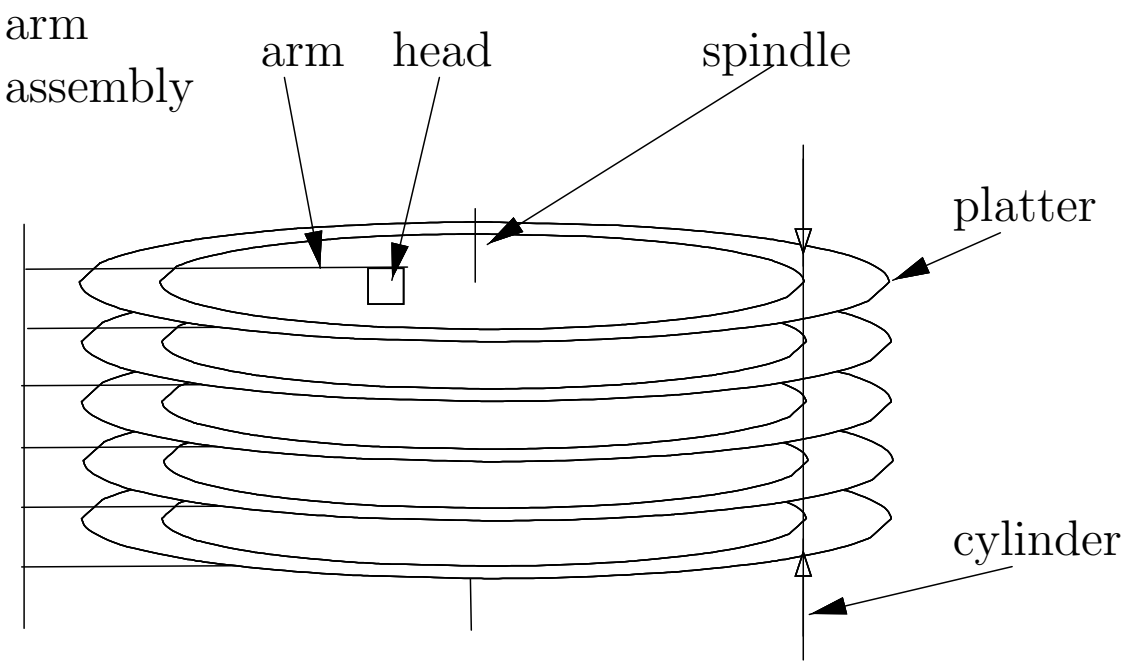
Goal:

- estimation and optimization of disk access costs

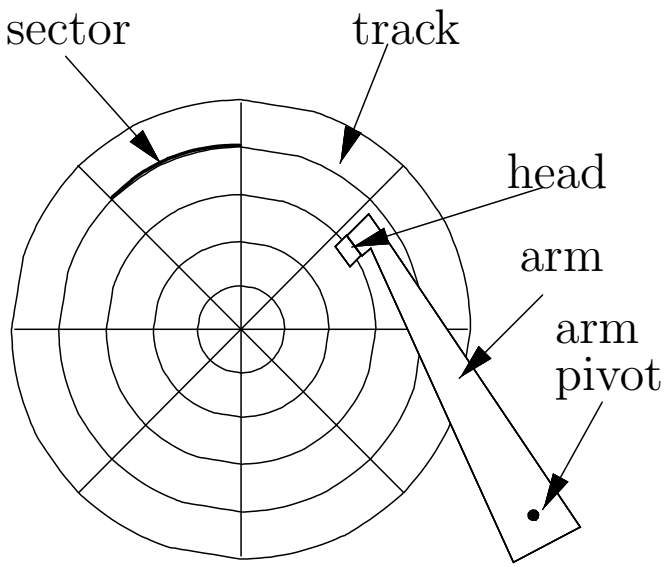
4. Accessing the Data (2)

- disk drives
- database buffer
- physical database organization
- physical algebra
- temporal relations and table functions
- indices
- counting the number of accesses
- disk drive costs
- selectivity estimations

Assembly



a. side view



b. top view

Zones

- outer tracks/sectors longer than inner ones
- highest density is fixed
- results in waste in outer sectors
- thus: cylinders organized into zones

Zones (2)

- every zone contains a fixed number of consecutive cylinders
- every cylinder in a zone has the same number of sectors per track
- outer zones have more sectors per track than inner zones
- since rotation speed is fixed: higher throughput on outer cylinders

Track Skew

Read all sectors of all tracks of some consecutive cylinders:

- read all sectors of one track
- switch to next track: small adjustment of head necessary
  called: *head switch*
- this causes tiny delay
- thus, if all tracks start at the same angular position then we miss the start of the first sector of the next track
- remedy: *track skew*

Cylinder Skew

Read all sectors of all tracks of some consecutive cylinders:

- read all sectors of all tracks of some cylinder
- switching to the next cylinder causes some delay
- again, we miss the start of the first sector, if the tracks start all start at the same angular position
- remedy: *cylinder skew*

Addressing Sectors

- physical Address: cylinder number, head (surface) number, sector number
- logical Address: LBN (logical block number)

LBN to Physical Address

Mapping:

Cylinder	Track	LBN	number of sectors per track
0	0	0	573
	1	573	573
...	...	...	...
	5	2865	573
1	0	3438	573
...	...	...	...
15041	0	35841845	253
...	...	...	...

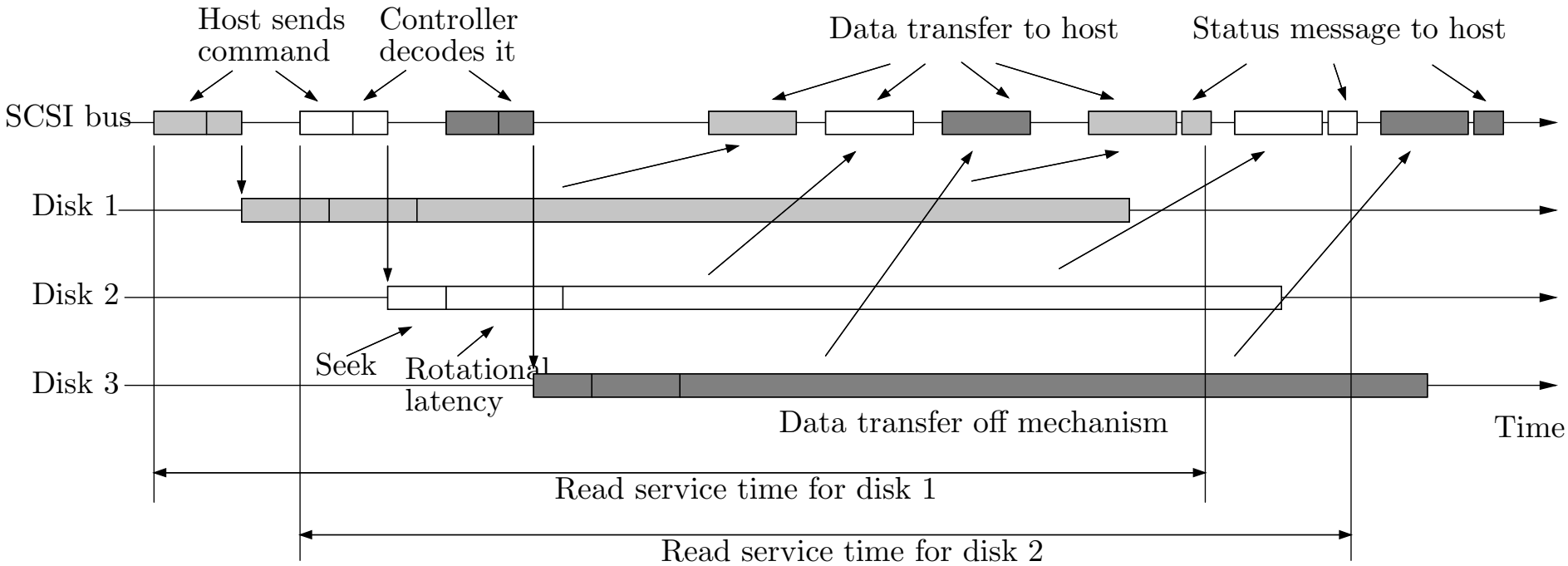
LBN to Physical Address (2)

This ideal view of the mapping is disturbed by *bad blocks*

- due to the high density, no perfect manufacturing is possible
- as a consequence *bad blocks* occur (sectors that cannot be used)
- reserve some blocks, tracks, cylinders for remapping bad blocks

Bad blocks may cause hickups during sequential reads

Reading/Writing a Block



Reading/Writing a Block (2)

1. the host sends the SCSI command.
2. the disk controller decodes the command and calculates the physical address.
3. during the seek the disk drive's arm is positioned such that the according head is correctly placed over the cylinder where the requested block resides. This step consists of several phases.
   - 3.1 the disk controler accelerates the arm.
   - 3.2 for long seeks, the arm moves with maximum velocity (coast).
   - 3.3 the disk controler slows down the arm.
   - 3.4 the disk arm settles for the desired location. The settle times differ for read and write requests. For reads, an aggressive strategy is used. If, after all, it turns out that the block could not be read correctly, we can just discard it. For writing, a more conservative strategy is in order.
4. the disk has to wait until the sector where the requested block resides comes under the head (rotation latency).
5. the disk reads the sector and transfers data to the host.
6. finally, it sends a status message.

Optimizing Round Trip Time

- caching
- read-ahead
- command queuing

Seek Time

A good approximation of the seek time where d cylinders have to be travelled is given by

$$seektime(d) = \begin{cases} c_1 + c_2\sqrt{d} & d \leq c_0 \\ c_3 + c_4 d & d > c_0 \end{cases}$$

where the constants c_i are disk specific. The constant c_0 indicates the maximum number cylinders where no coast takes place: seeking over a distance of more than c_0 cylinders results in a phase where the disk arm moves with maximum velocity.

Cost model: initial thoughts

Disk access costs depend on

- the current position of the disk arm and
- the angular position of the platters

Both are not known at query compilation time
Consequence:

- estimating the costs of a single disk access at query compilation time may result in large estimation error

Better: costs of many accesses
Nonetheless: First Simplistic Cost Model to give a feeling for disk drive access costs

Simplistic Cost Model

We introduce some disk drive parameters for out simplistic cost model:

- average latency time: average time for positioning (seek+rotational delay)
  - use average access time for a single request
  - Estimation error can (on the average) be as "low" as 35%
- sustained read/write rate:
  - after positioning, rate at which data can be delivered using sequential read

Model 2004

A hypothetical disk (inspired by disks available in 2004) then has the following parameters:

Model 2004		
Parameter	Value	Abbreviated Name
capacity	180 GB	D_{cap}
average latency time	5 ms	D_{lat}
sustained read rate	100 MB/s	D_{srr}
sustained write rate	100 MB/s	D_{swr}

The time a disk needs to read and transfer n bytes is then approximated by $D_{\text{lat}} + n/D_{\text{srr}}$.

Sequential vs. Random I/O

Database management system developers distinguish between

- *sequential* I/O and
- *random* I/O.

In our simplistic cost model:

- for sequential I/O, there is only one positioning at the beginning and then, we can assume that data is read with the sustained read rate.
- for random I/O, one positioning for every unit of transfer—typically a page of say 8 KB—is assumed.

Simplistic Cost Model

Read 100 MB

- Sequential read: 5 ms + 1 s
- Random read (8K pages): 65 s

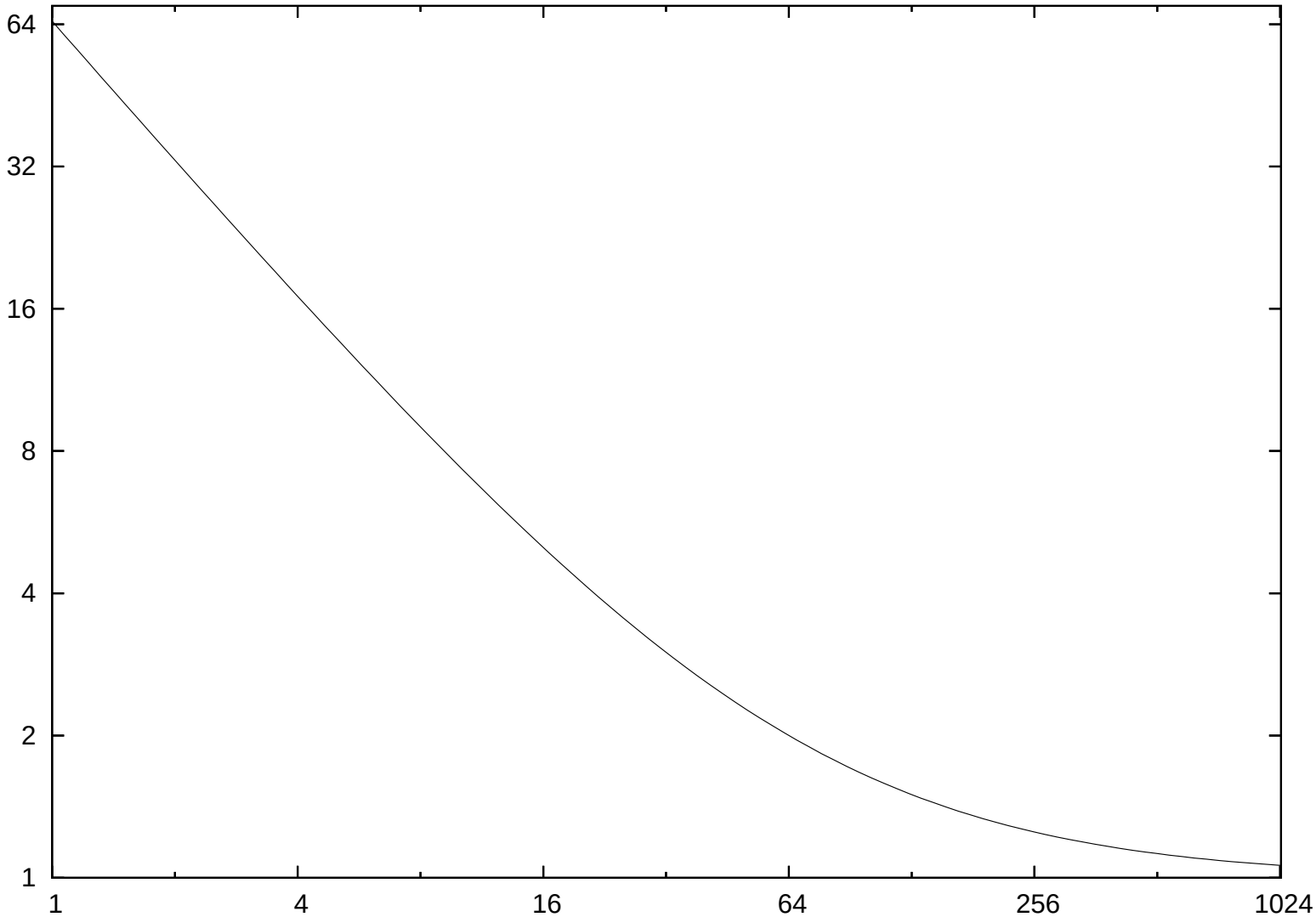
Simplistic Cost Model (2)

Problems:

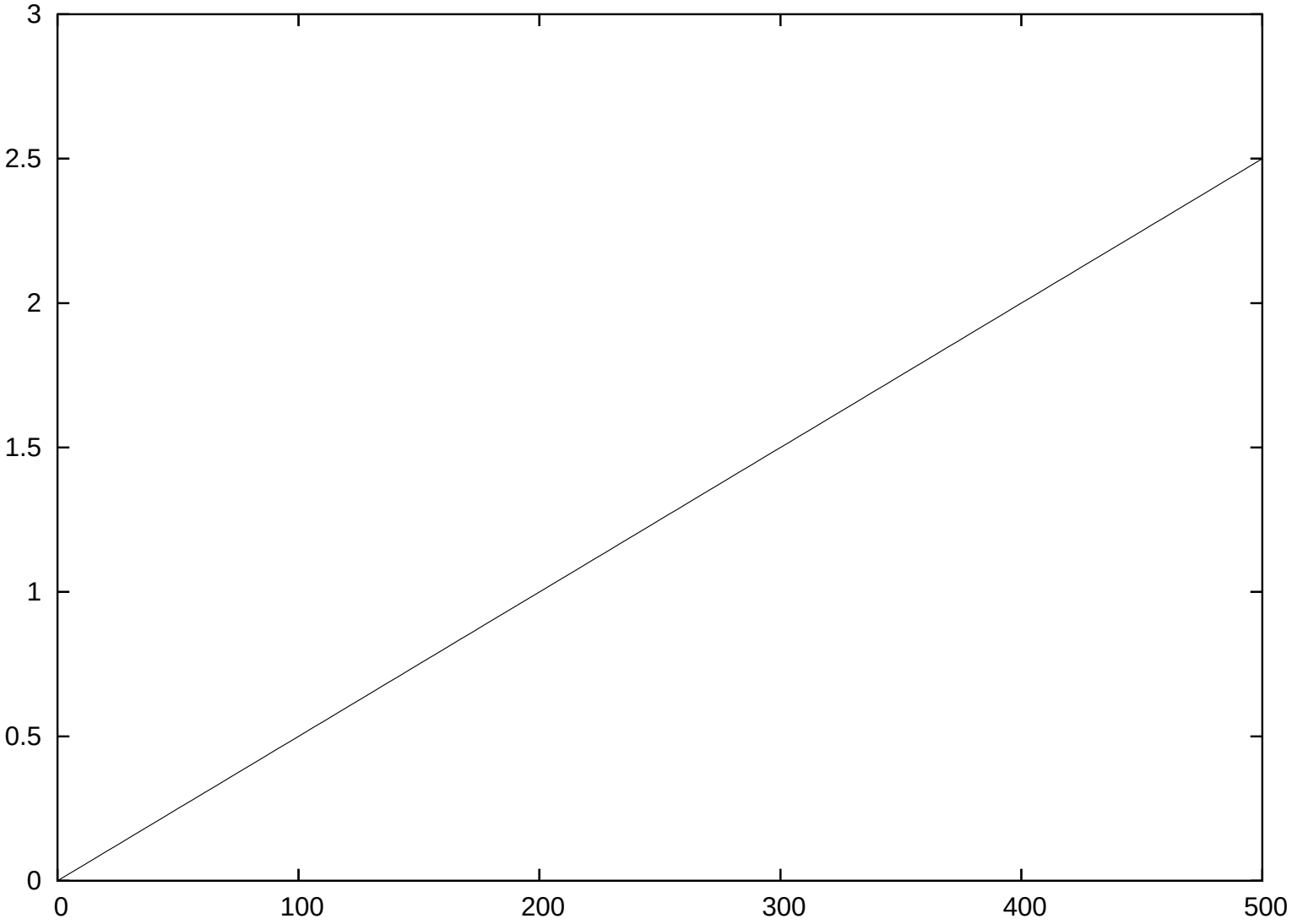
- other applications
- other transactions
- other read operations in the same QEP

may request blocks from the same disk and move away the head(s) from the current position
Further: 100 MB sequential search poses problem to buffer manager

Time to Read 100 MB (x: number of 8 KB chunks)



Time to Read n Random Pages



Simplistic Cost Model (3)

100 MB can be stored on 12800 8 KB pages.
In our simplistic cost model, reading 200 pages randomly costs about the same as reading 100 MB sequentially.
That is, reading 1/64th of 100 MB randomly takes as long as reading the 100 MB sequentially.

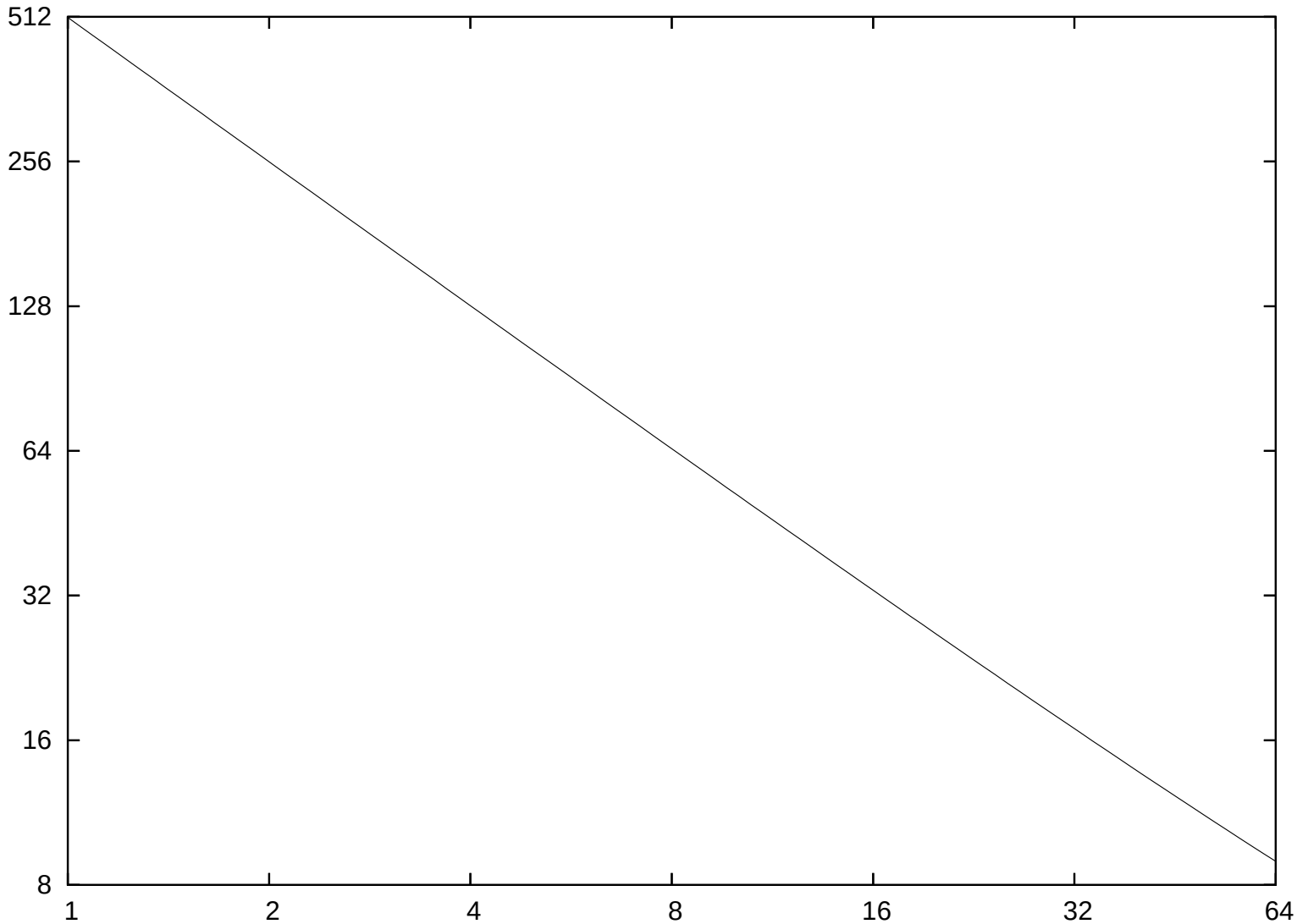
Simplistic Cost Model (4)

Let us denote by a the positioning time, s the sustained read rate, p the page size, and d some amount of consecutively stored bytes. Let us calculate the break even point

$$\begin{aligned} n * (a + p/s) &= a + d/s \\ n &= (a + d/s)/(a + p/s) \\ &= (as + d)/(as + p) \end{aligned}$$

a and s are disk parameters and, hence, fixed. For a fixed d, the break even point depends on the page size.
Next Figure: x-axis: is the page size p in multiples of 1 K; y-axis: $(d/p)/n$ for $d = 100$ MB.

Break Even Point (depending on page size)



Two Lessons Learned

- sequential read is much faster than random read
- the runtime system should secure sequential read

The latter point can be generalized:

- the runtime system of a database management system has, as far as query execution is concerned, two equally important tasks:
  - allow for efficient query evaluation plans and
  - allow for smooth, simple, and robust cost functions.

Measures to Achieve the Above

Typical measures on the database side are

- carefully chosen physical layout on disk
  (e.g. cylinder or track-aligned extents, clustering)
- disk scheduling, multi-page requests
- (asynchronous) prefetching,
- piggy-back scans,
- buffering (e.g. multiple buffers, replacement strategy) and last but not least
- efficient and robust algorithms for algebraic operators