BOSTON UNIVERSITY

CS 561: Data Systems Architectures

class 2

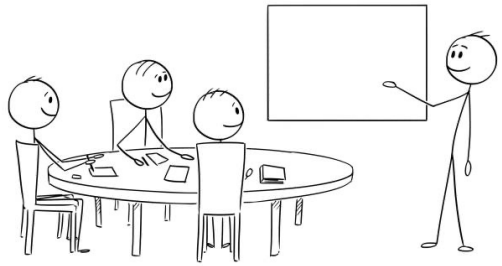
Data Systems 101

Tarikul Islam Papon

https://bu-disc.github.io/CS561/

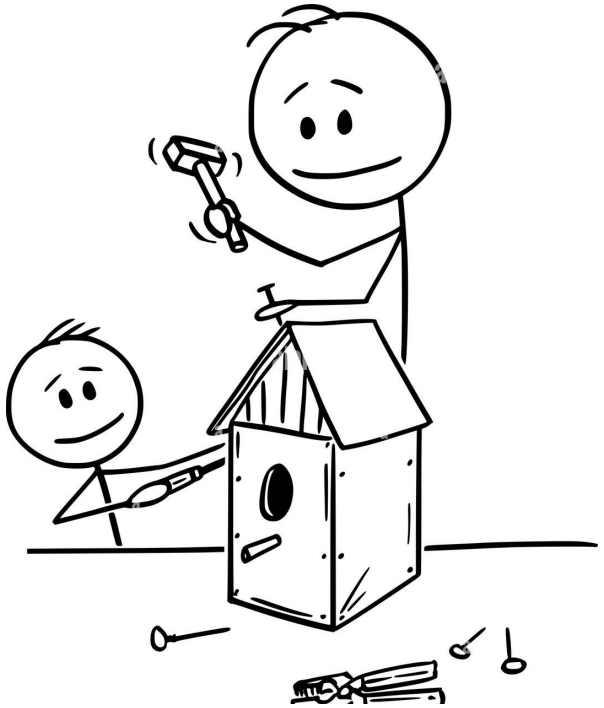
What do we do in this class?

reading papers



presentations

reviews



projects

BOSTON UNIVERSITY

Projects

project 0

A small implementation project
to sharpen dev skills

<u>independent project</u>

Due on Feb 2, 2024

AND

project 1

A medium project to give you a flavor of
large-scale production system

<u>groups of 3</u>

BOSTON
UNIVERSITY

Projects

AND

project 0

A small implementation project
to sharpen dev skills

<u>independent project</u>

Due on Feb 2, 2024

project 1

A medium project to give you a flavor of
large-scale production system

<u>groups of 3</u>

Start forming groups
Due on Feb 16, 2024

Projects

OR

systems project

groups of 2/3

implementation-heavy C/C++ project

research project

groups of 3

pick a subject (list will be available soon)

design & analysis

experimentation

BOSTON UNIVERSITY

Projects

OR

systems project

groups of 3

implementation-heavy C/C++ project

1. Proposal
2. Mid-semester report
3. Final report + Presentation

research project

groups of 3

pick a subject (list available in website)

design & analysis

experimentation

class timeline

discussions
interaction in OH & Lab
questions

Week 2
register for presentations by 1/31

now

Week 3
project 0 by 2/2
form groups by 2/4

first paper review +
first paper presentation 2/7

Week 5
project 1 by 2/16

Week 6
submit project proposal by 2/23

Week 10
submit mid-semester
project report by 3/22

Week 15: Project presentations
submit all material by 4/26

Piazza

2 classes per week & OH/Labs 5 days per week

all discussions & announcements

http://piazza.com/bu/spring2022/cs561/

also available on class website

We have added everyone who already registered!

Please double-check!

BOSTON UNIVERSITY

size (volume)

rate (velocity)

sources (variety)

veracity & value

big data

(it's not only about size)

The 3 V's

BOSTON UNIVERSITY

size (volume)

rate (velocity)

sources (variety)

veracity & value

big data

(it's not only about size)

The 3 V's

+ our ability to collect ***machine-generated*** data

 scientific experiments

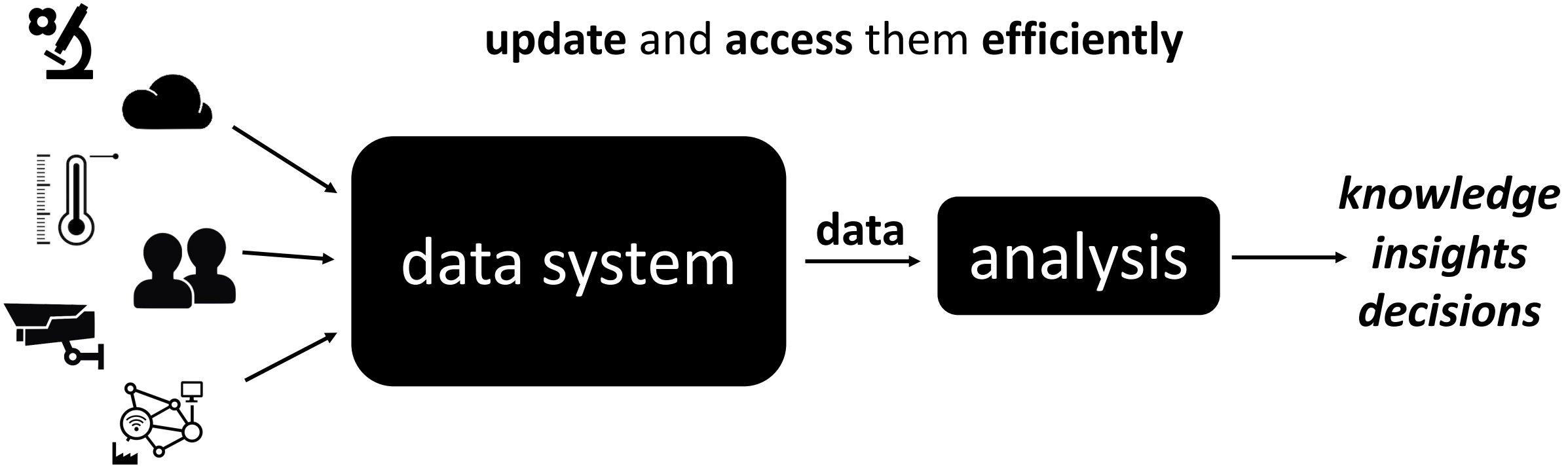
 sensors

social 

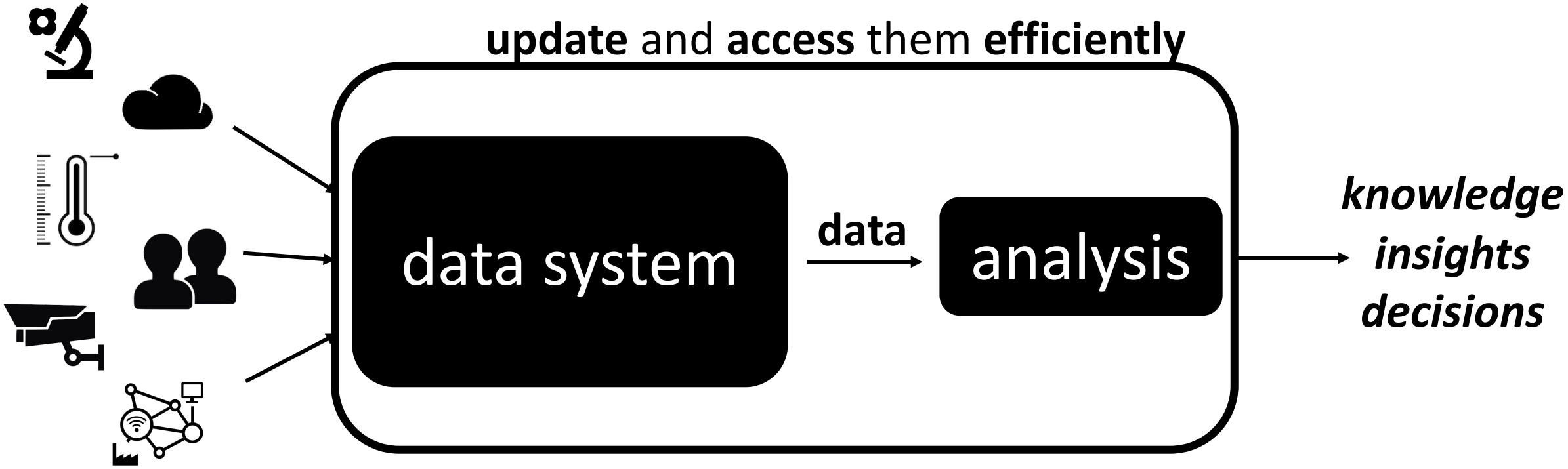
Internet-of-things 

BOSTON UNIVERSITY

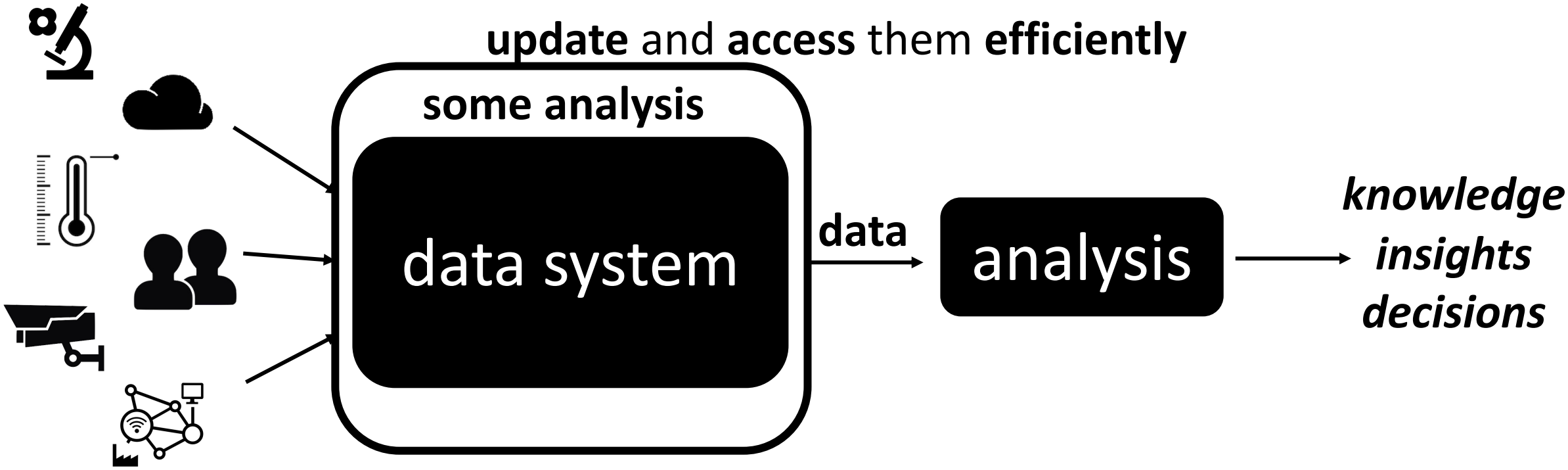
a **data system** is a large software system
that **stores data**, and provides the **interface** to
update and **access** them **efficiently**



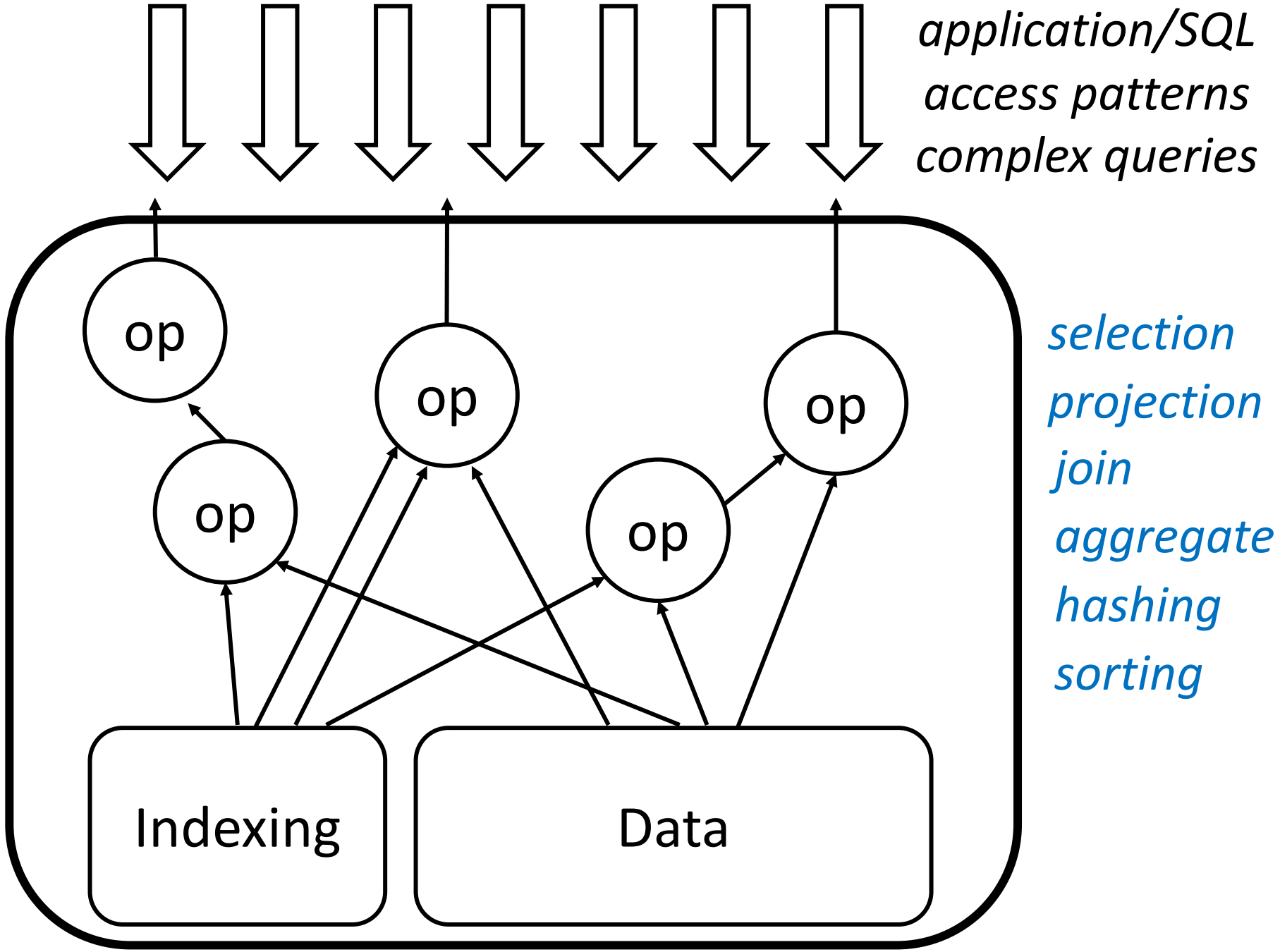
a **data system** is a large software system
that **stores data**, and provides the **interface** to
update and **access** them **efficiently**



a **data system** is a large software system
that **stores data**, and provides the **interface** to
update and **access** them **efficiently**



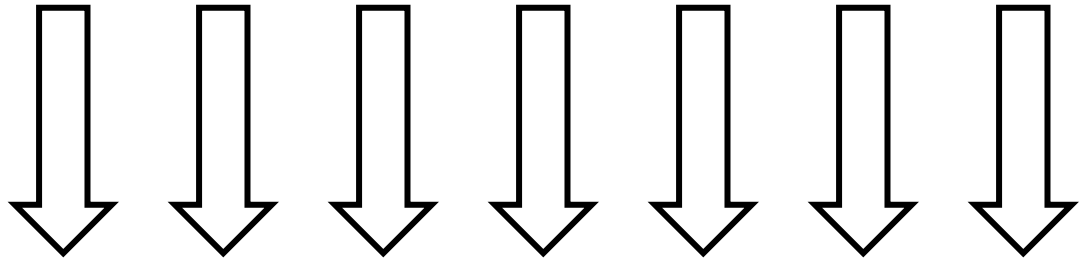
data system: breaking the blackbox



algorithms & operators

data & metadata

selection
projection
join
aggregate
hashing
sorting



application/SQL
access patterns
complex queries

modules

Query Parser

Query Compiler

Optimizer

Evaluation Engine

Memory/Storage Management

Indexing

Transaction Management

memory hierarchy

CPU

Caches

Memory

Disk

growing environment

DB

ACID
large systems
complex
lots of tuning

noSQL

BASE
simple, clean
“just enough”

>$200B by 2020, growing at 11.7% every year

[The Forbes, 2016]

growing environment

DB

ACID
large systems
complex
lots of tuning

noSQL

BASE
simple, clean
“just enough”

>$200B by 2020, growing at 11.7% every year

[The Forbes, 2016]

$3B by 2020, growing at 20% every year

[Forrester, 2016]

growing environment

DB

ACID
large systems
complex
lots of tuning

noSQL

BASE
simple, clean
"just enough"

more **complex** applications

need for **scalability**

>$200B by 2020, growing at 11.7% every year

[The Forbes, 2016]

newSQL

$3B by 2020, growing at 20% every year

[Forrester, 2016]

BOSTON UNIVERSITY

growing environment

DB

ACID
large systems
complex
lots of tuning

noSQL

BASE
simple, clean
"just enough"

more **complex** applications

need for **scalability**

>$200B by 2020, growing at 11.7% every year

[The Forbes, 2016]

newSQL

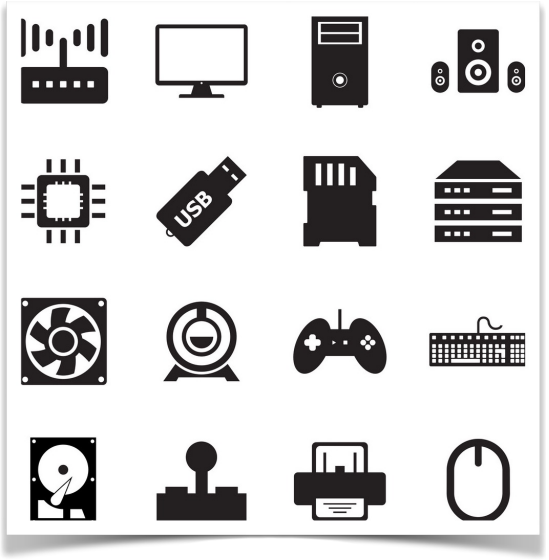
$3B by 2020, growing at 20% every year

[Forrester, 2016]

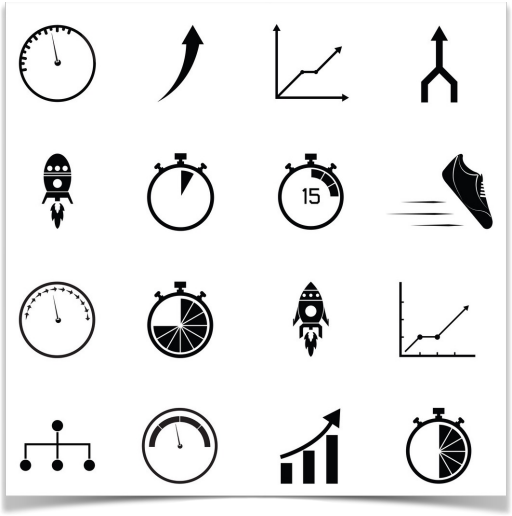
growing need for tailored systems

more data

new hardware



new applications



new performance goals

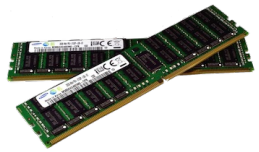
data systems & the hardware

memory hierarchy

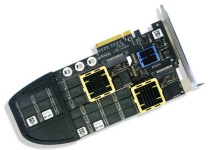
CPU

on-chip cache

on-board cache

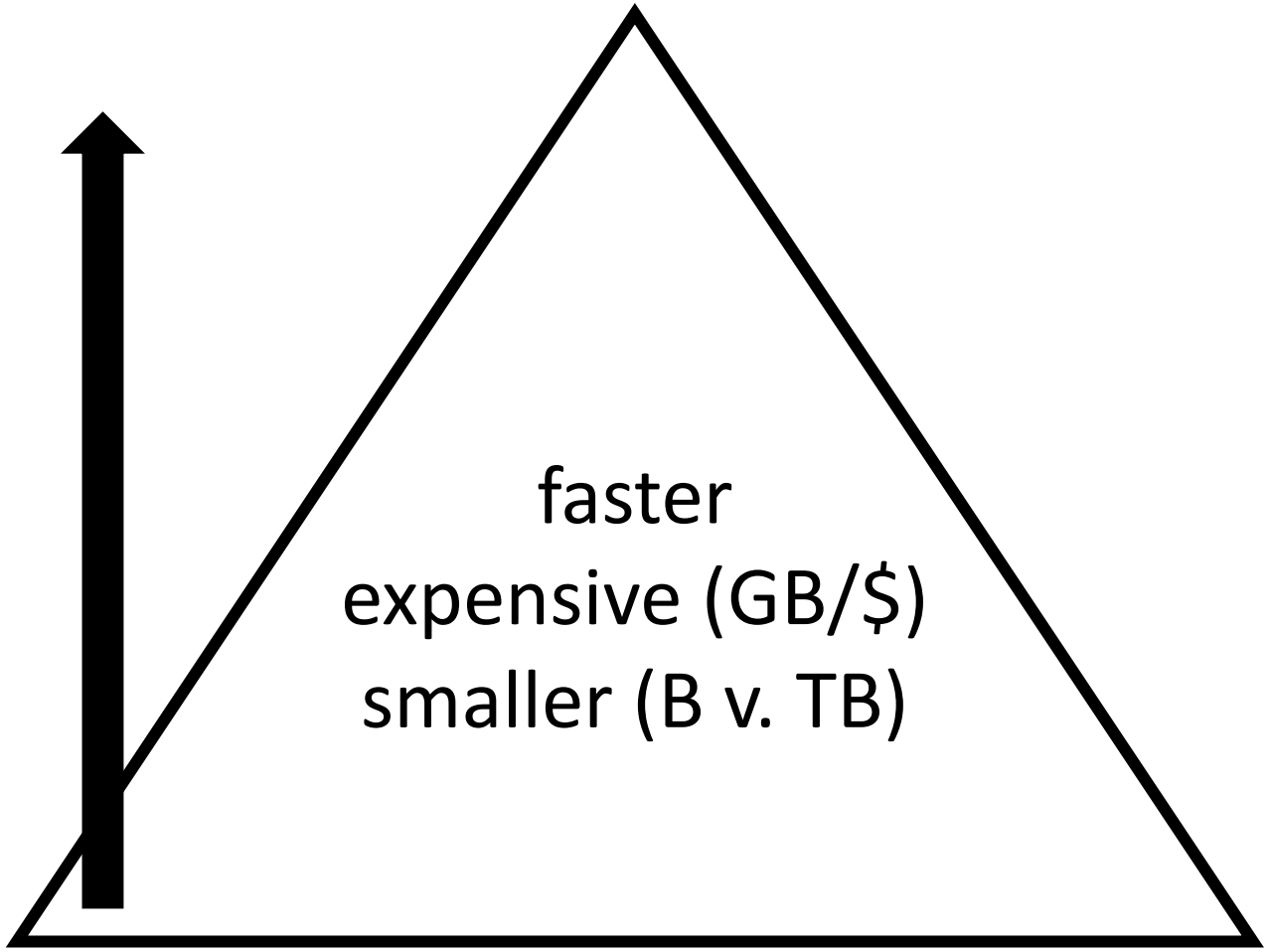


main memory

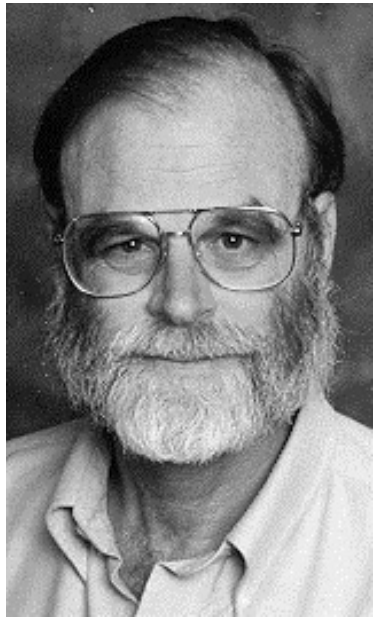


flash storage

magnetic disks



memory hierarchy (by Jim Gray)



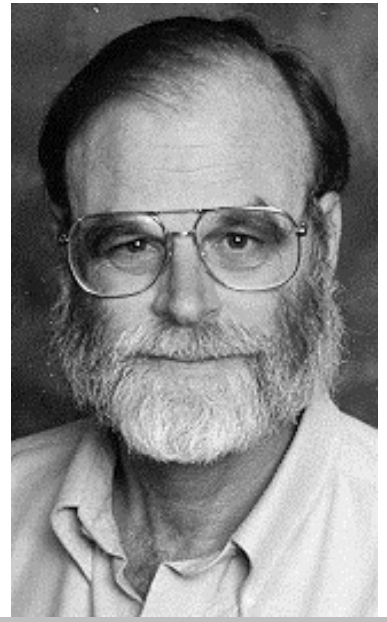
Jim Gray, IBM, Tandem, Microsoft, DEC

ACM Turing Award 1998

ACM SIGMOD Edgar F. Codd Innovations award 1993

	registers/CPU	my head ~0
2x	**on chip cache**	this room 1min
10x	**on board cache**	this building 10min
100x	**memory**	Washington, DC 5 hours
10^6x	**disk**	Pluto 2 years
10^9x	**tape**	Andromeda 2000 years

memory hierarchy (by Jim Gray)

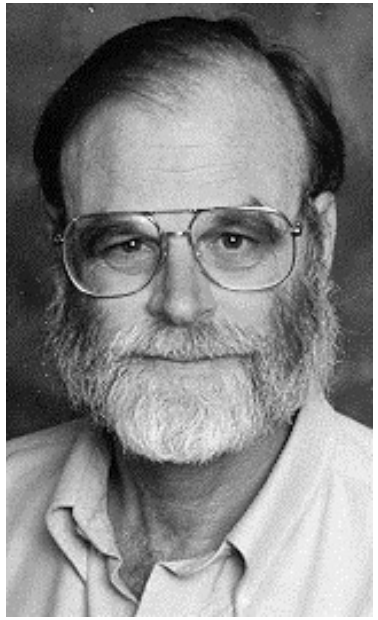


	registers/CPU	my head ~0
2x	**on chip cache**	this room 1min
10x	**on board cache**	this building 10min

45TB @ $150

tape?
sequential-only magnetic storage
still a multi-billion industry

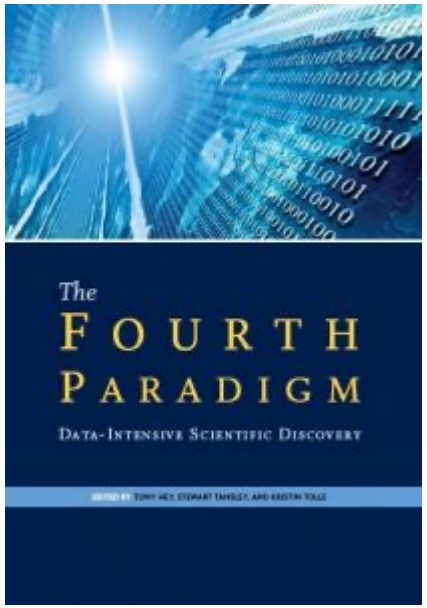
Jim Gray (a great scientist and engineer)



Jim Gray, IBM, Tandem, Microsoft, DEC

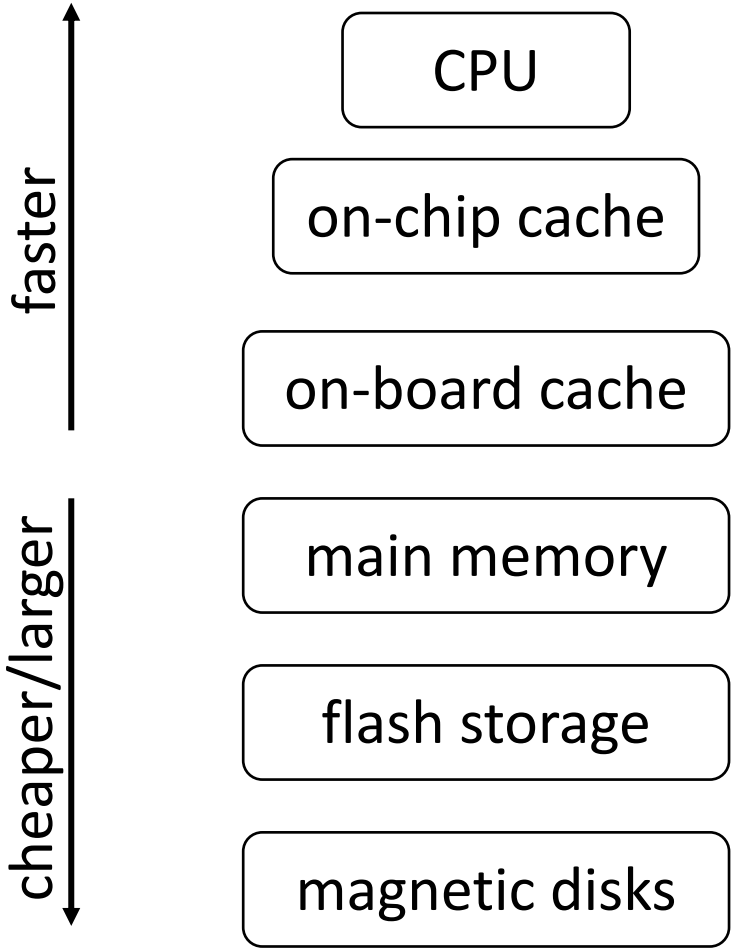
ACM Turing Award 1998

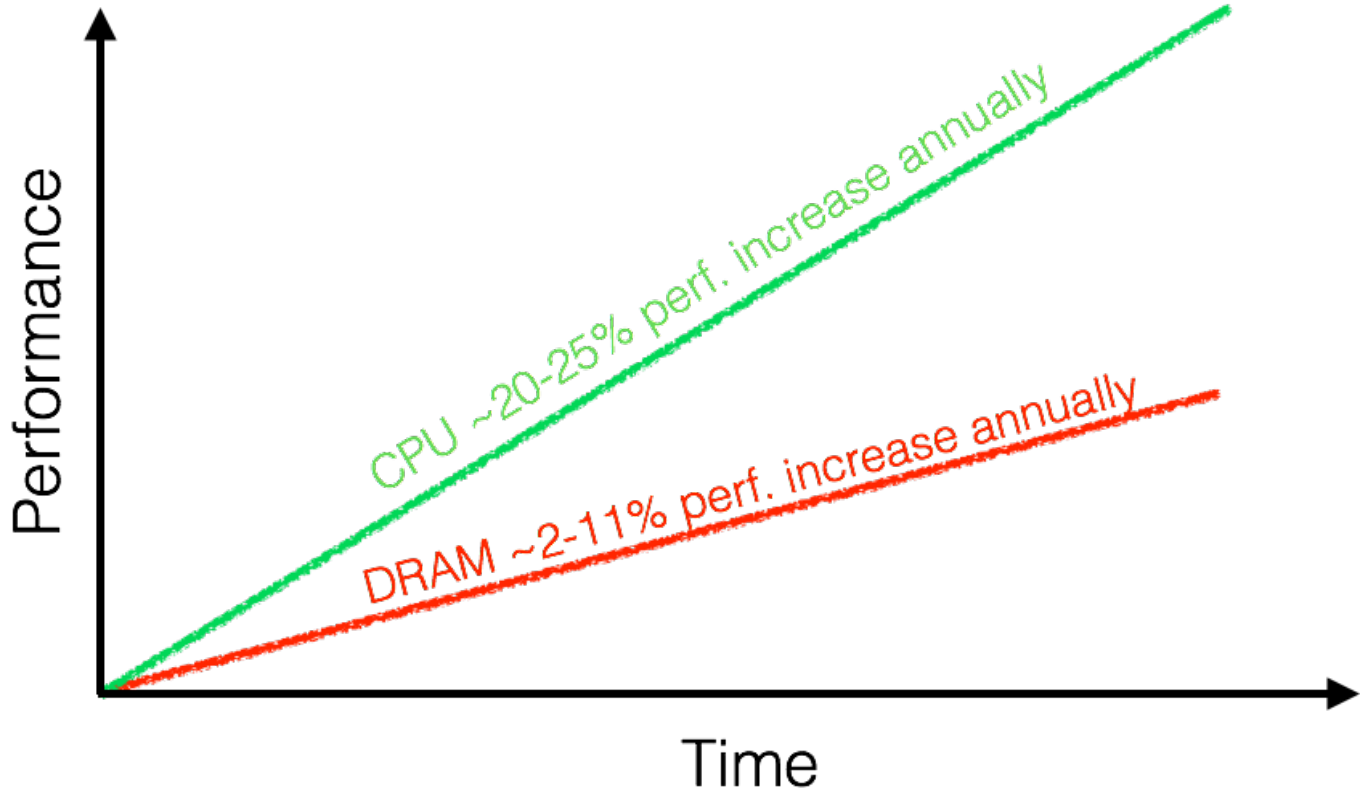
ACM SIGMOD Edgar F. Codd Innovations award 1993



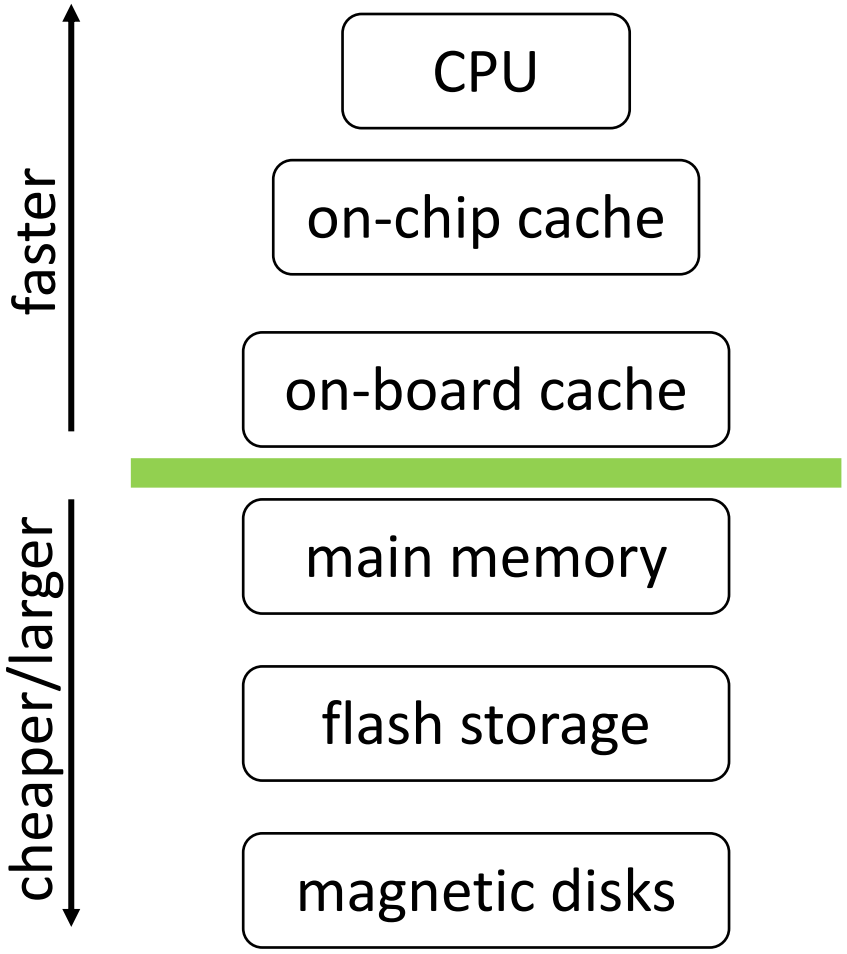
the first collection of
technical visionary research on
a data-intensive scientific discovery

memory wall

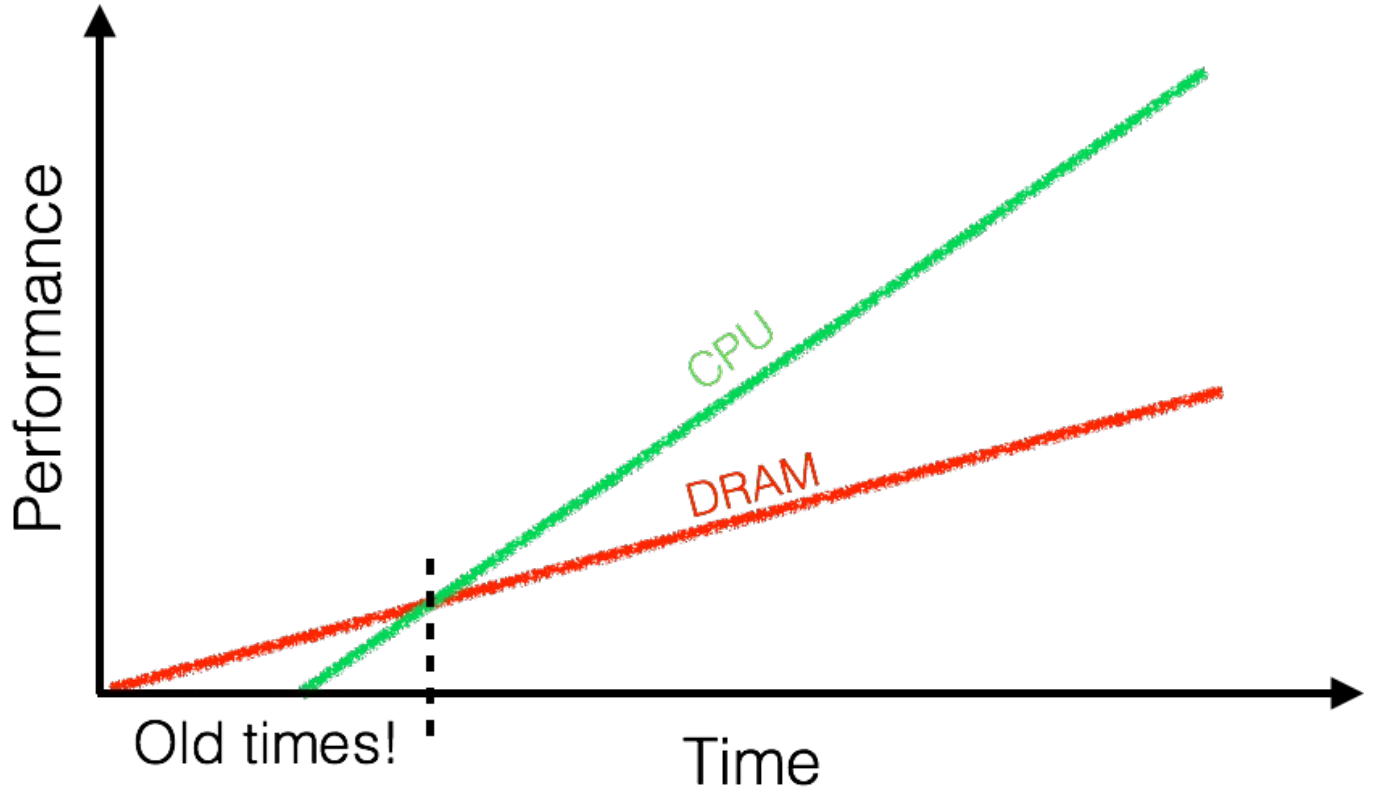




memory wall

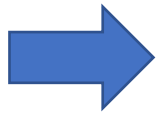


be careful when you go below the green line



cache/memory misses

computations happen here



CPU

on-chip cache

on-board cache

main memory

flash storage

magnetic disks

be careful when you go below the green line

cache miss: looking for something that is not in the cache

what happens if I miss?

memory miss: looking for something that is not in memory

what happens if I miss again?

be very careful when you go below the green line

data movement

CPU

on-chip cache

on-board cache

main memory

flash storage

magnetic disks

data goes through
all necessary levels

also read
unnecessary data

Photo by Gary Dineen/NBAE via Getty Images

need to read only X
read the whole page

data movement

CPU

on-chip cache

on-board cache

main memory

flash storage

magnetic disks

data goes through
all necessary levels

also read
unnecessary data

Photo by Gary Dineen/NBAE via Getty Images

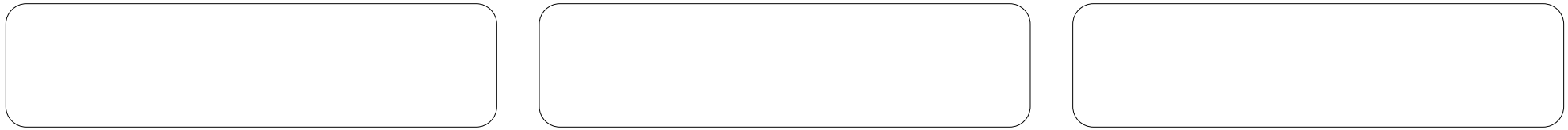
need to read only X
read the whole page

remember!

disk is millions (mem, hundreds) of times slower than CPU

page-based access & random access

query x<7



size=120 bytes

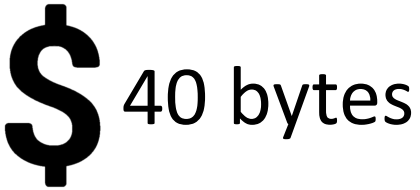
memory (memory level N)

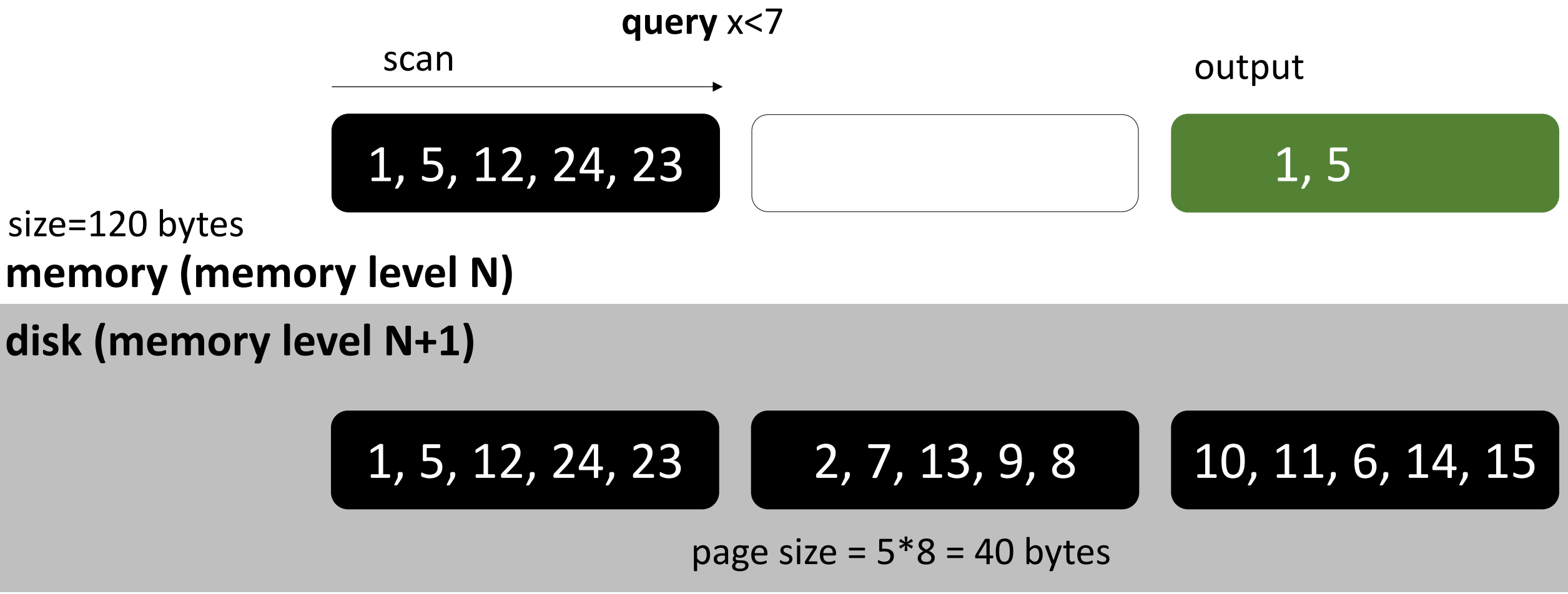
disk (memory level N+1)

1, 5, 12, 24, 23 | 2, 7, 13, 9, 8 | 10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

page-based access & random access





40 bytes

page-based access & random access

query x<7

scan

output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5

size=120 bytes

memory (memory level N)

disk (memory level N+1)

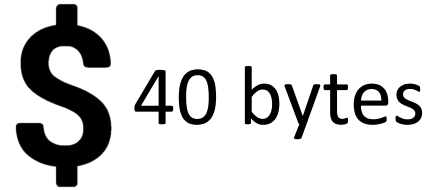
1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

page-based access & random access



query x<7

scan

output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

page-based access & random access

query x<7

scan

output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

page-based access & random access

query x<7

scan

output

10, 11, 6, 14, 15

2, 7, 13, 9, 8

1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

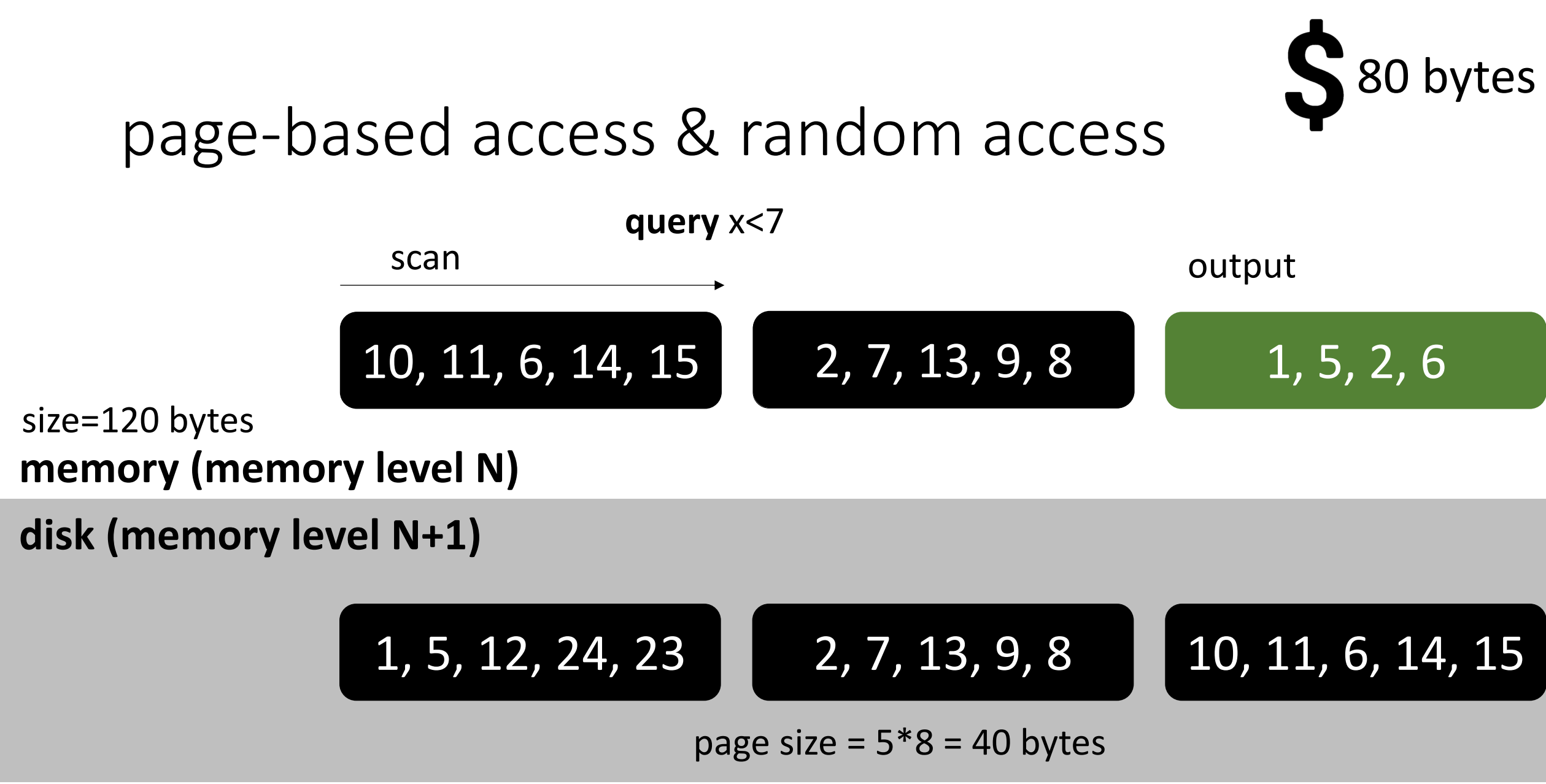
1, 5, 12, 24, 23

2, 7, 13, 9, 8

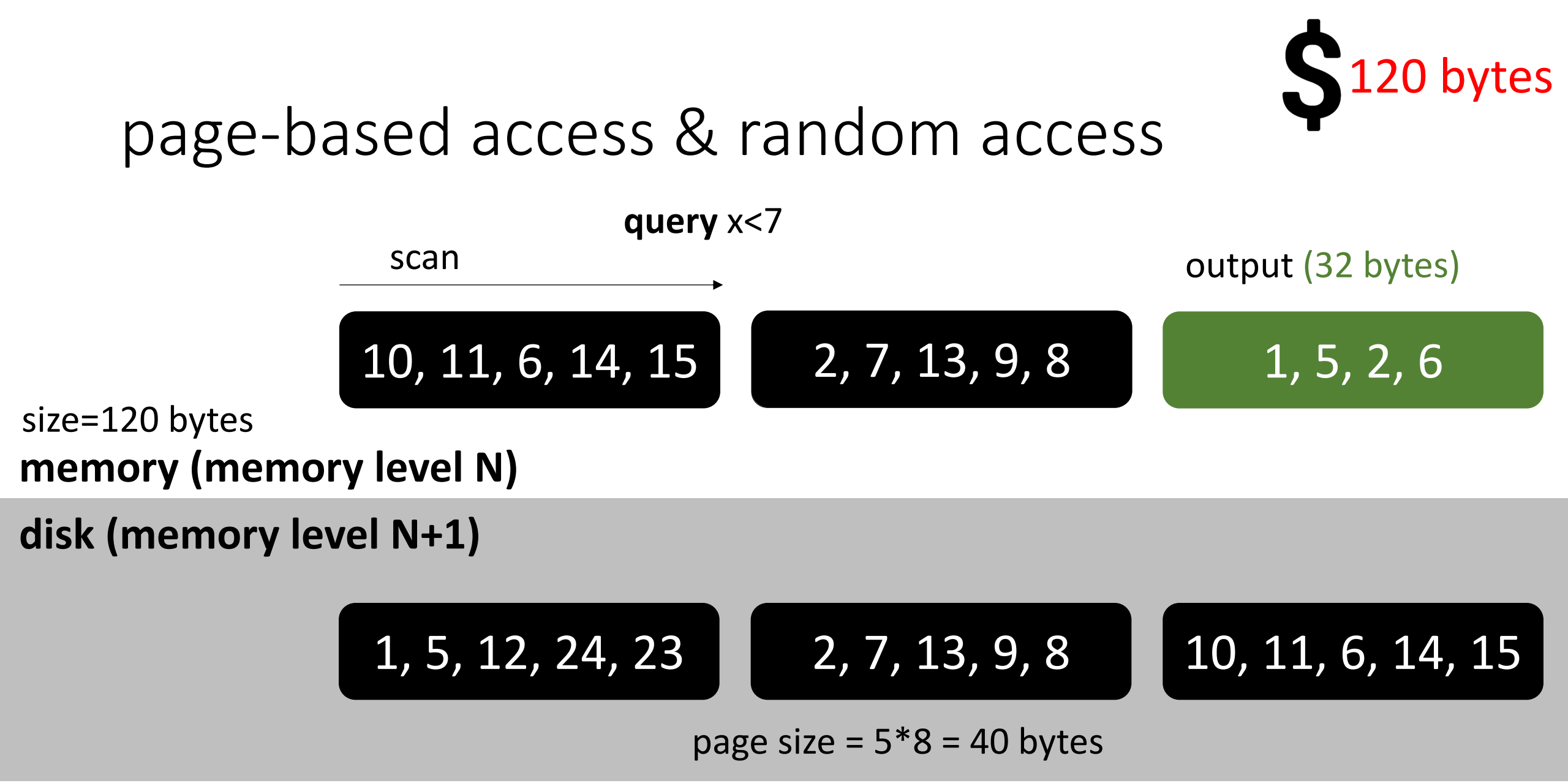
10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

page-based access & random access



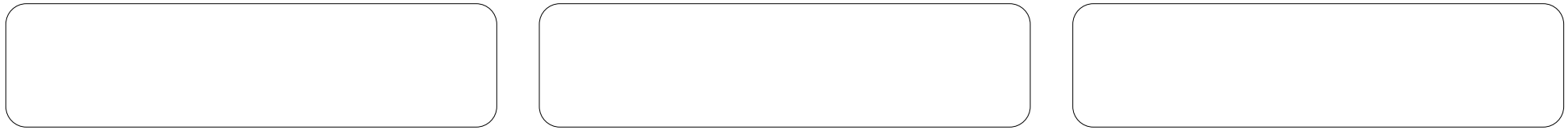
page-based access & random access



what if we had an oracle (perfect index)?

page-based access & random access

query x<7



size=120 bytes

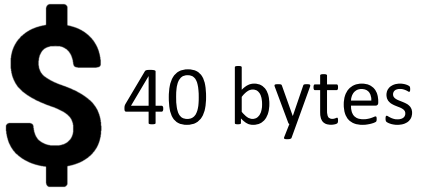
memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23 | 2, 7, 13, 9, 8 | 10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

page-based access & random access



query x<7

oracle

output

1, 5, 12, 24, 23

1, 5

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

page-based access & random access

query x<7

oracle

output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

page-based access & random access

40 bytes

query x<7

oracle

output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

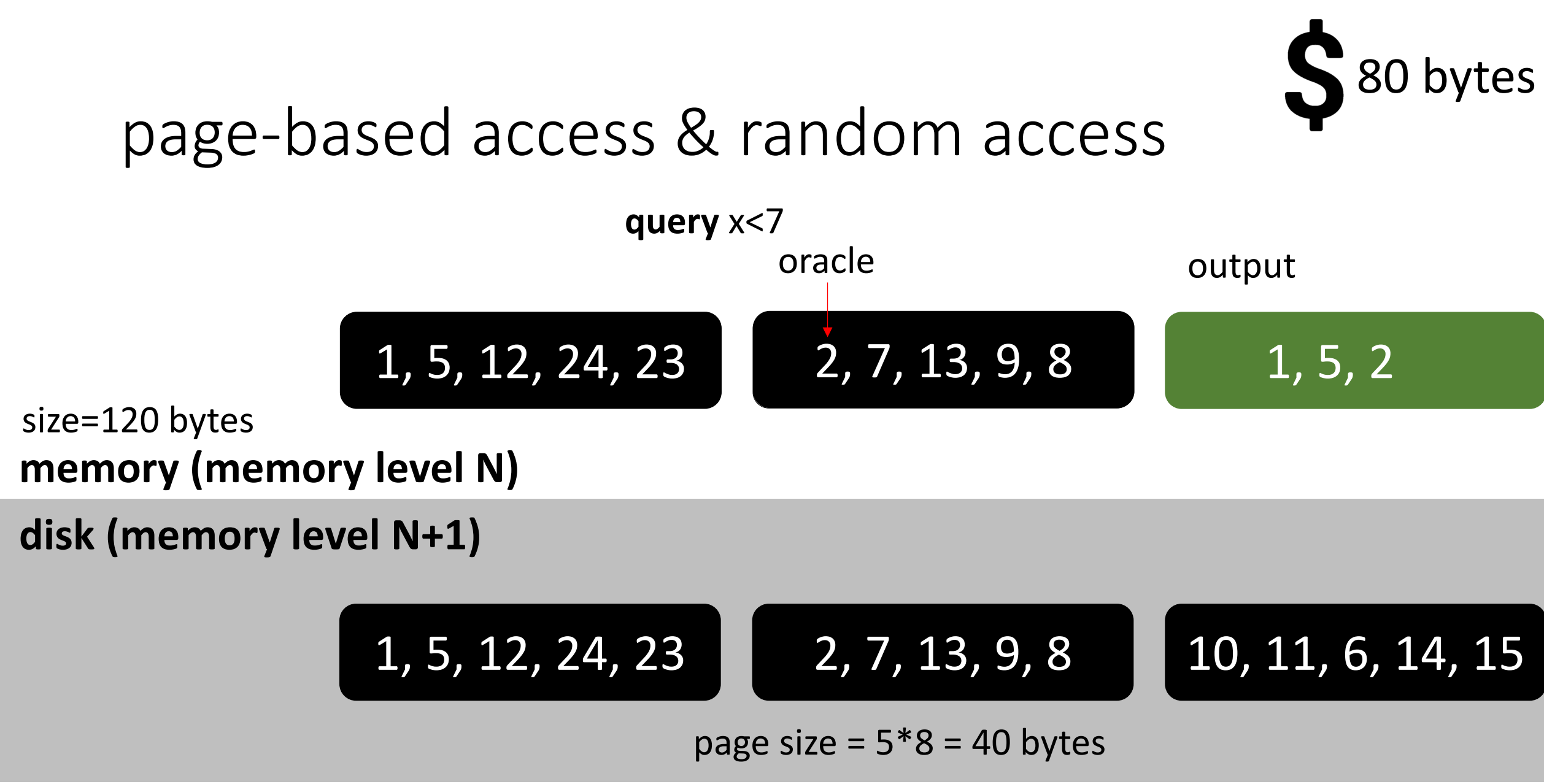
1, 5, 12, 24, 23

2, 7, 13, 9, 8

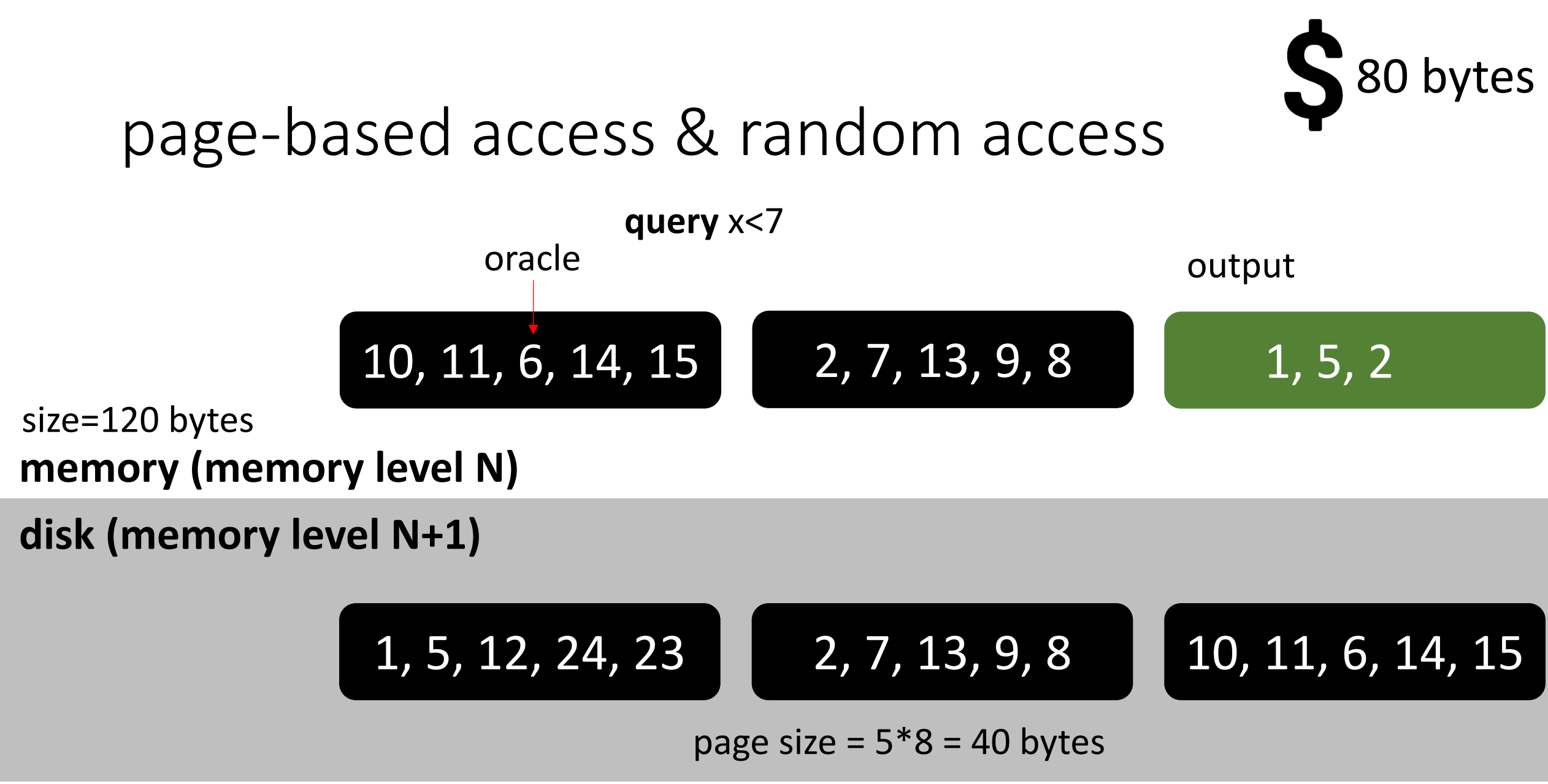
10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

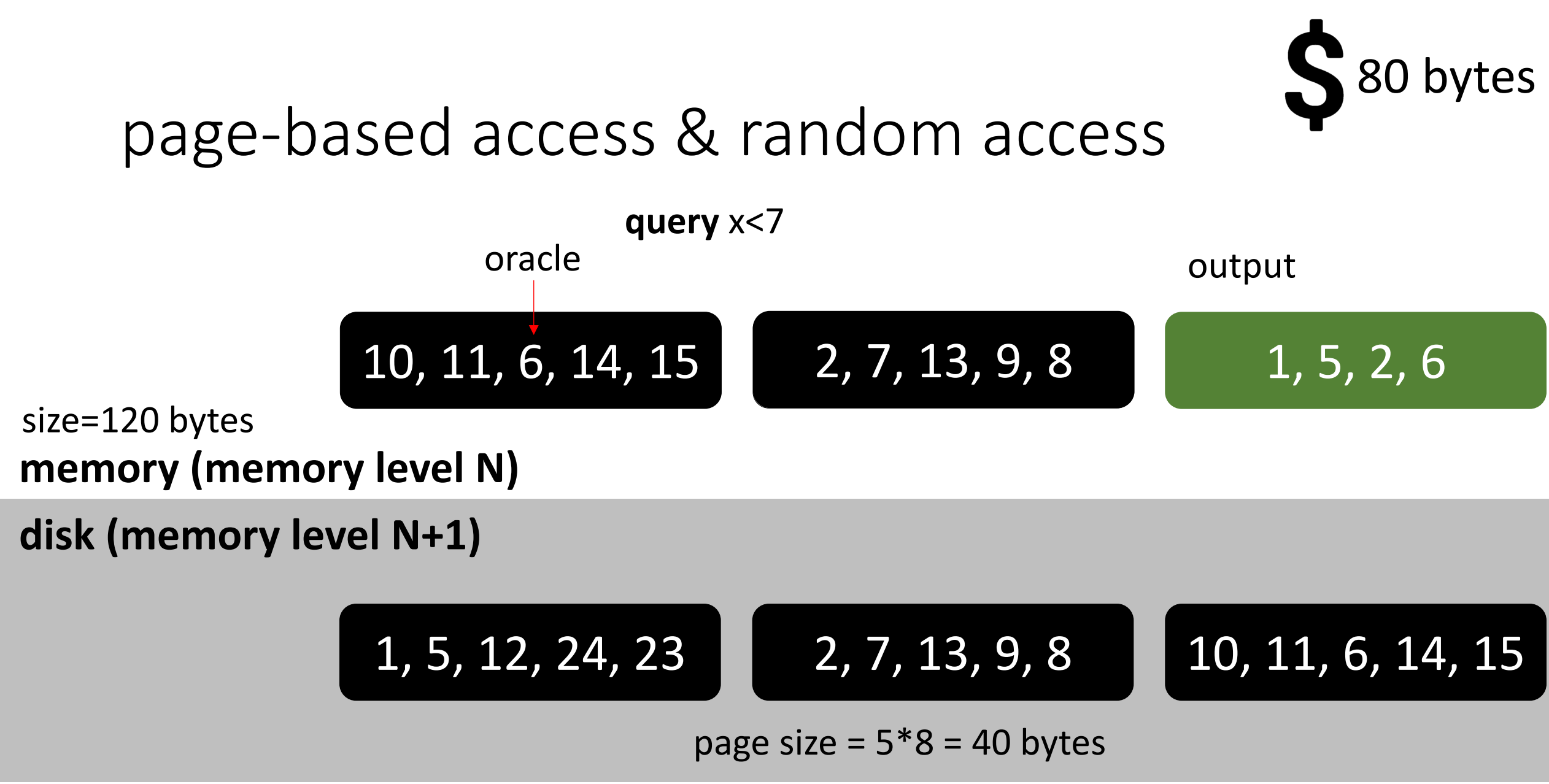
page-based access & random access



page-based access & random access



page-based access & random access



page-based access & random access

$ 120 bytes

was the oracle helpful?

query x<7

oracle

output (32 bytes)

10, 11, 6, 14, 15

2, 7, 13, 9, 8

1, 5, 2, 6

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

when is the oracle helpful?

for which query would an oracle help us?

how to decide whether to use the oracle or not?

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

BOSTON UNIVERSITY

every **byte** counts

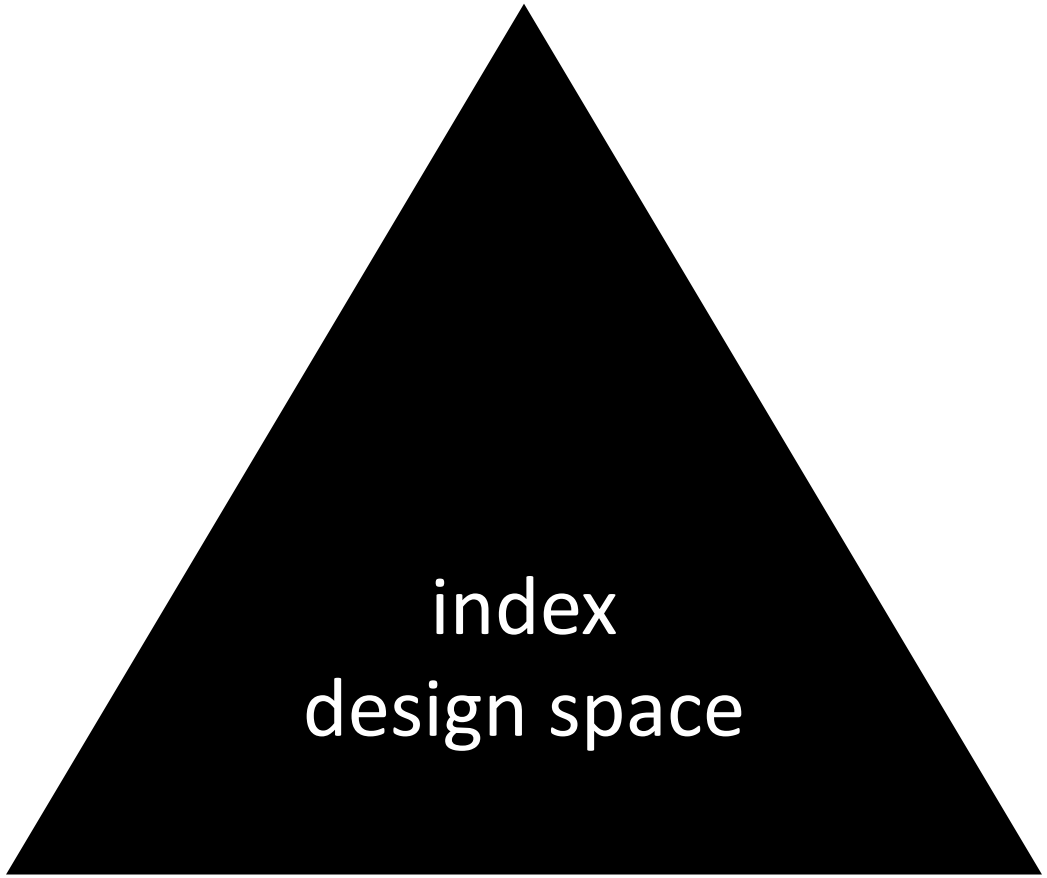
overheads and tradeoffs

how we store data

layouts, indexes

know the **query**

access path selection



rules of thumb

sequential access

read one block; consume it completely; discard it; read next

hardware can predict and start prefetching

prefetching can exploit full memory/disk bandwidth

random access

read one block; consume it partially; discard it; (may re-use)

are random accesses always bad?

the one that helps us **avoid a large number of accesses** (random or sequential)

zonemaps

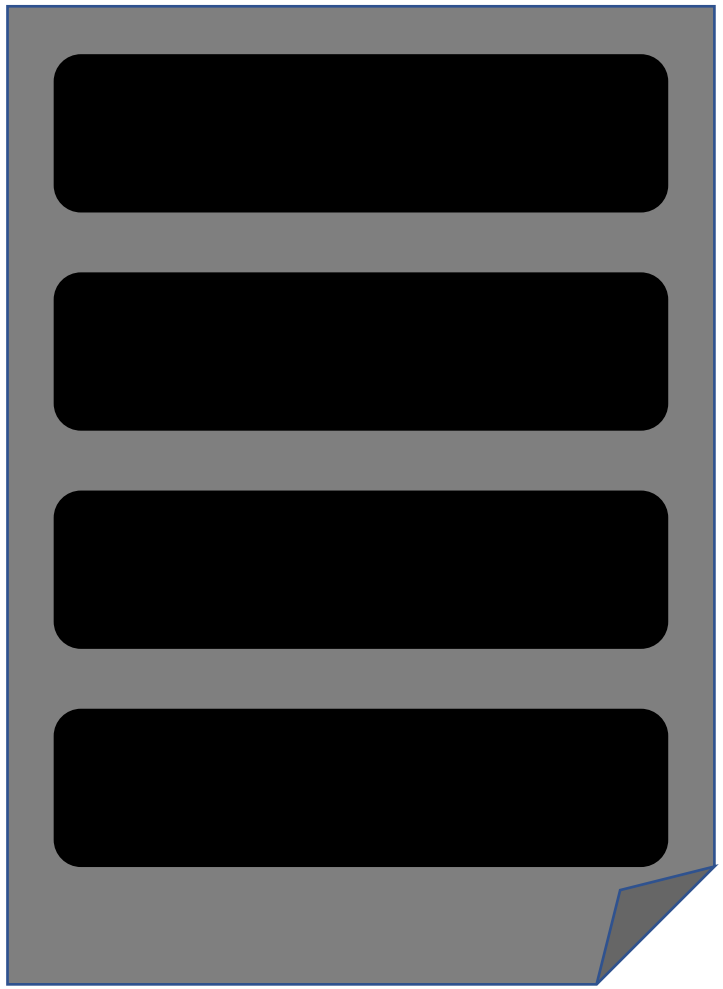
file = collection of pages

page 0

page 1

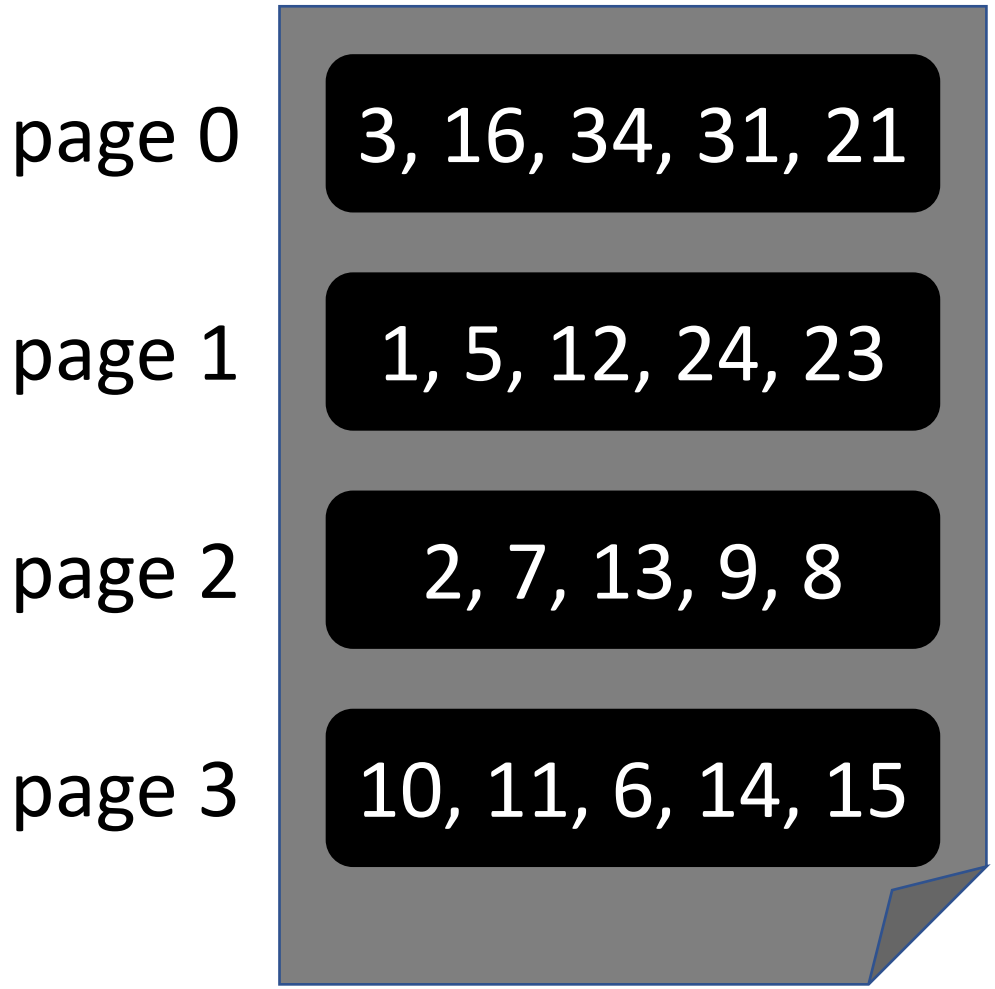
page 2

page 3



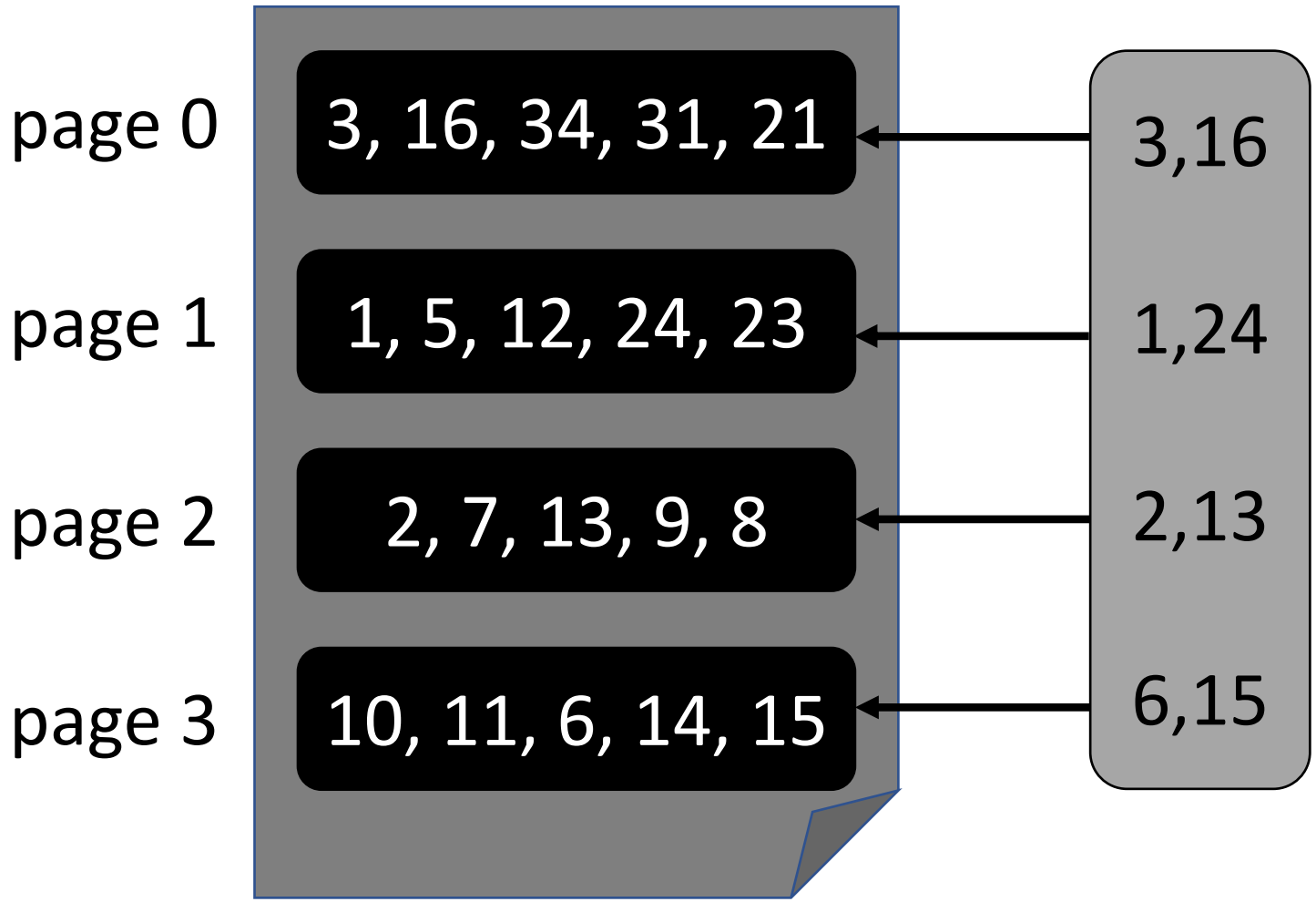
zonemaps

file = collection of pages



zonemaps

file = collection of pages



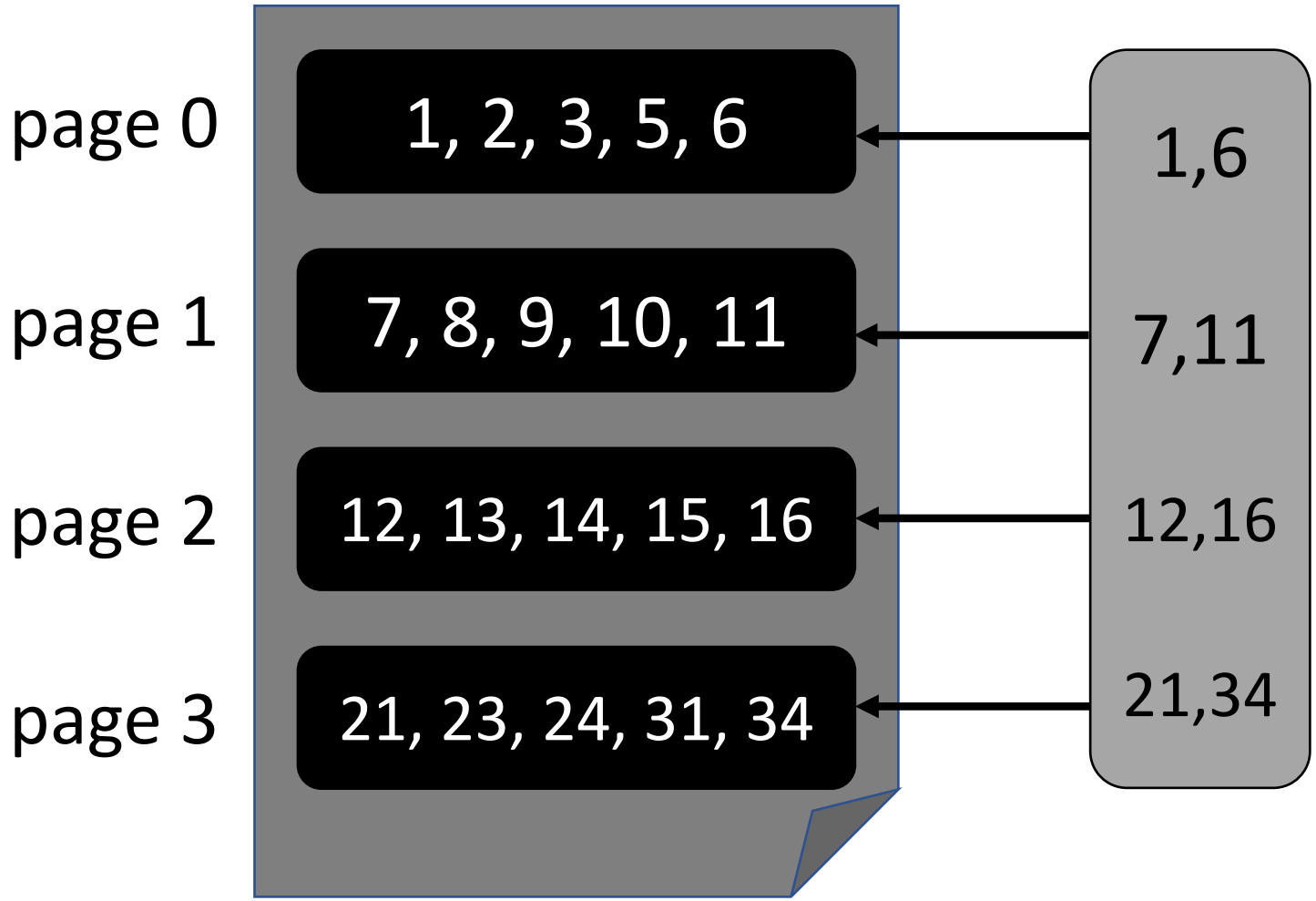
light-weight

typically retained in memory

But what if the data is sorted?

zonemaps

file = collection of pages



light-weight

typically retained in memory

But what if the data is sorted?

the language of efficient systems: C/C++

why?

fewer assumptions

low-level control over hardware

make decisions about physical data placement and consumptions

the language of efficient systems: C/C++

why?

fewer assumptions

we want you in the project to make low-level decisions

BOSTON UNIVERSITY

CS 561: Data Systems Architectures

class 2

Data Systems 101

next : modern main-memory data systems & semester project