

COSI 167A Advanced Data Systems

Class 2

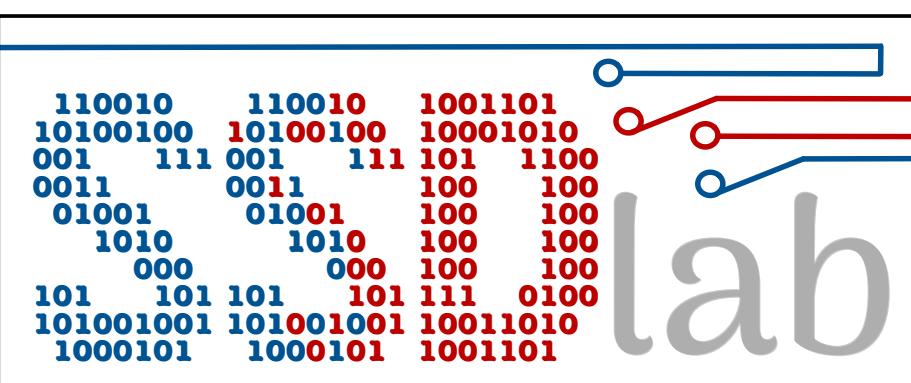
Data Systems 101

Prof. Subhadeep Sarkar



Brandeis
UNIVERSITY

<https://ssd-brandeis.github.io/COSI-167A/>



Today in COSI 127B

What's on the cards?

class logistics, goals, and administrivia

introduction to NoSQL systems

Project 1 details

Recap: Why take the class?

Introduction to “modern” databases!

BIG data

Data-driven world, Unstructured data

store and manage data

Data is generated at an unprecedented rate and volume
—“Does your system SCALE?”

querying BIG data & querying **fast**

Querying unstructured data, SQL?

new system designs

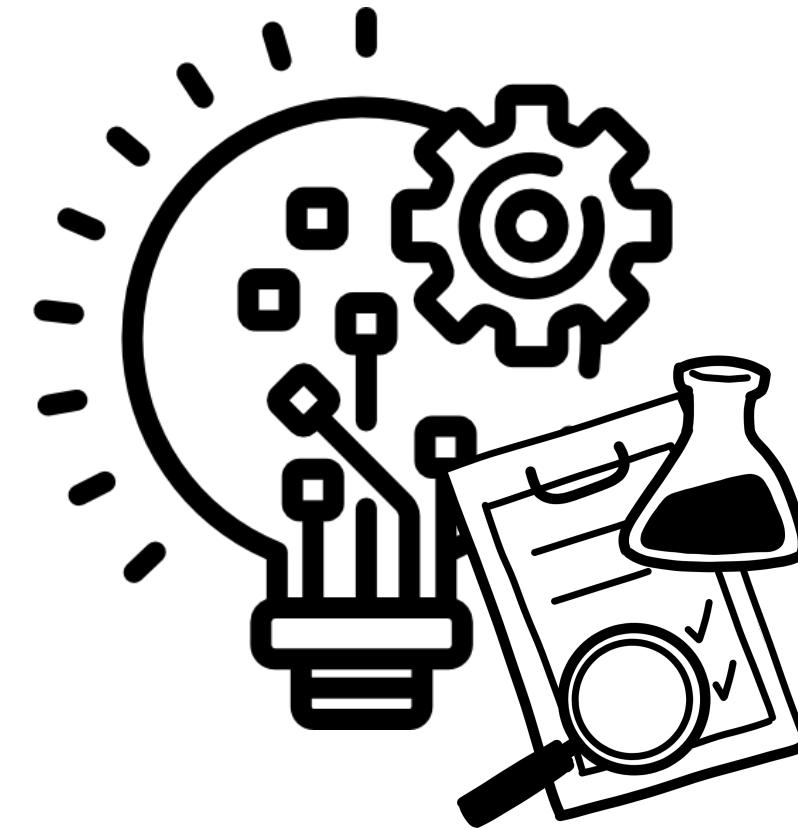
New application requirements = New design trends

getting your **hands “dirty”**

Play with large-scale, commercial storage engines

Recap: Class **Philosophy**

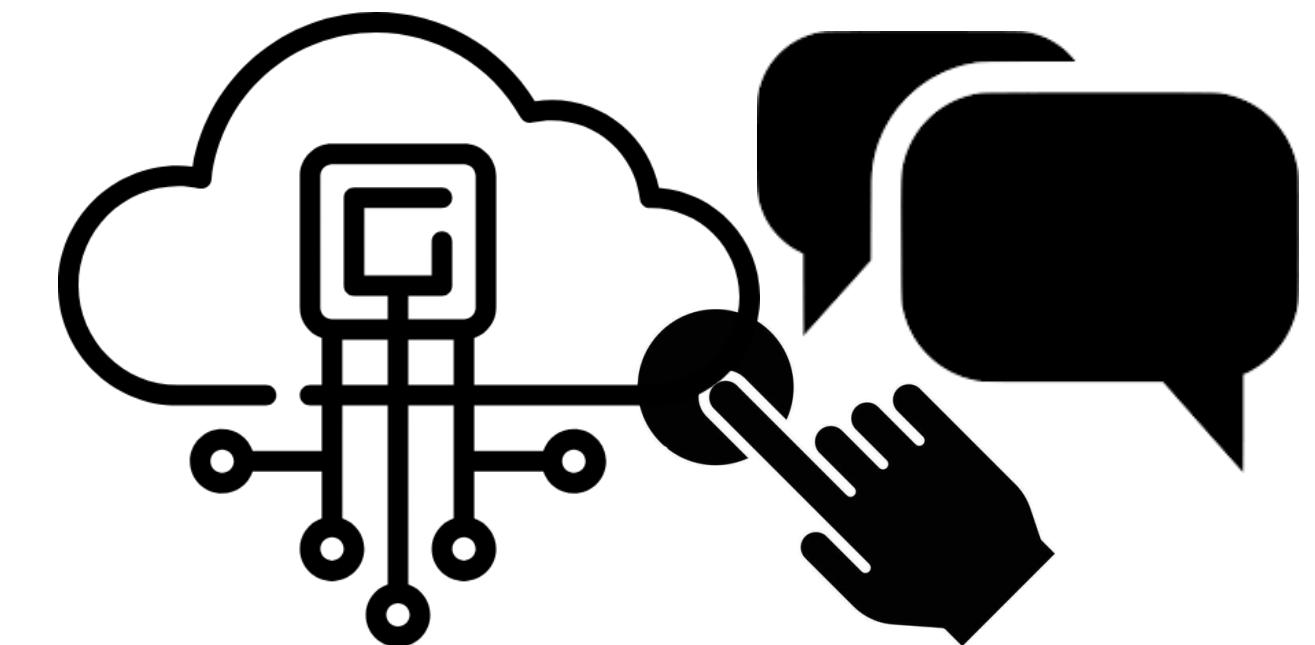
Principles to go by



cutting-edge
research



question
everything



interactive &
collaborative

Learn to *think out of the box!*

Recap: Readings

Papers, papers, and papers



Architecture of a Database System

— J. Hellerstein, M. Stonebraker and J. Hamilton

Foundations and Trends in Databases, 2007

Chapters 1, 3, 5

The Design and Implementation of Modern Column-store Database Systems

— D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden

Foundations and Trends in Databases, 2013

Chapters 1, 2, 3

Data Structures for Data-Intensive Applications

— Manos Athanassoulis, Stratos Idreos, and Dennis Shasha,

Foundations and Trends in Databases, 2023.

Semester reading!

Evaluation

A guide to success!

class participation

5%

in-class participation, SH visits



Ask questions!

& answer my questions!

goal: learning through discussion

There's **NO** stupid question!

Evaluation

A guide to success!

class participation

5%

in-class participation, SH visits

paper reviews

9%

3 reviews during the semester

goal: learn to **parse** and **critique** state-of-the-art research papers

no more than **one page**; more details before first review

Tuning NoSQL storage engines
ML and systems
Indexing

Evaluation

A guide to success!

class participation

5%

in-class participation, SH visits

paper reviews

9%

3 reviews during the semester

technical questions

16%

8 technical questions during the semester

goal: understand the **core concepts** presented in a paper

1-2 paragraphs; be concise and to the point

Evaluation

A guide to success!

class participation

5%

in-class participation, SH visits

paper reviews

9%

3 reviews during the semester

technical questions

16%

8 technical questions during the semester

paper presentation

15%

2 students per presentation

goal: learn to **present** technical papers & prepare **slides**

you choose the paper; **registration link to open soon!**

Evaluation

A guide to success!

class participation

5%

in-class participation, SH visits

paper reviews

9%

3 reviews during the semester

technical questions

16%

8 technical questions during the semester

paper presentation

15%

2 students per presentation

projects

55%

Project 1 + Class project

Evaluation

A guide to success!

class participation

5%

in-class participation, SH visits

paper reviews

9%

3 reviews during the semester

technical questions

16%

8 technical questions during the semester

paper presentation

15%

2 students per presentation

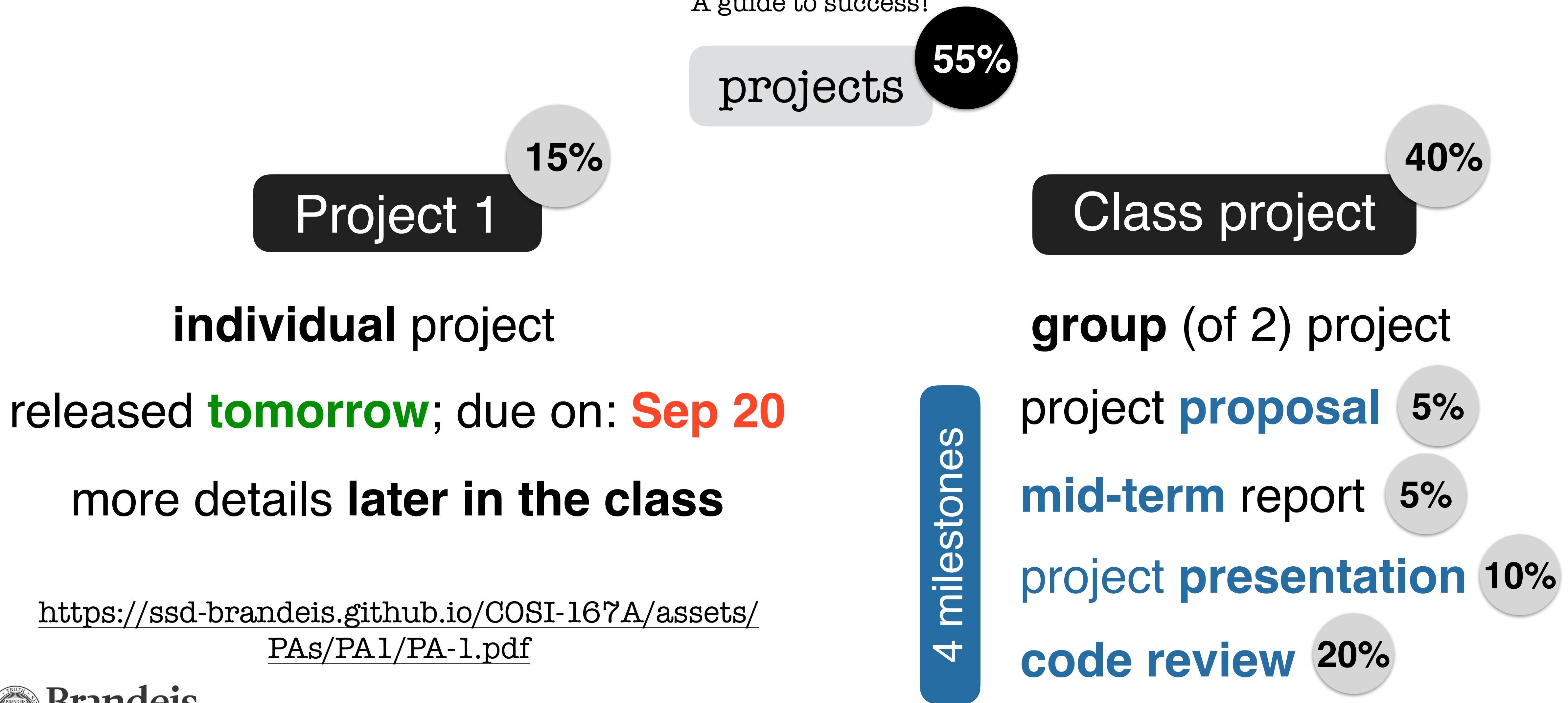
projects

55%

Project 1 + Class project

Evaluation

A guide to success!



Data is **everywhere!**

Everyone produces data!



experimental physics (IceCube, CERN)
neuroscience
biology

data mining business datasets
machine learning and AI for corporate and consumer



online gaming
micro-transactions, economics

Data is **everywhere!**

Everyone produces data!



experimental physics (IceCube, CERN)
neuroscience
biology

machine learning and AI for corporate and consumer



online gaming
micro-transactions, economics



How much data is generated **every day** in 2024? 

500 TB

Data is **everywhere!**

Everyone produces data!



experimental physics (IceCube, CERN)
neuroscience
biology

machine learning and AI for corporate and consumer



online gaming
micro-transactions, economics



How much data is generated **every day** in 2024? ☰

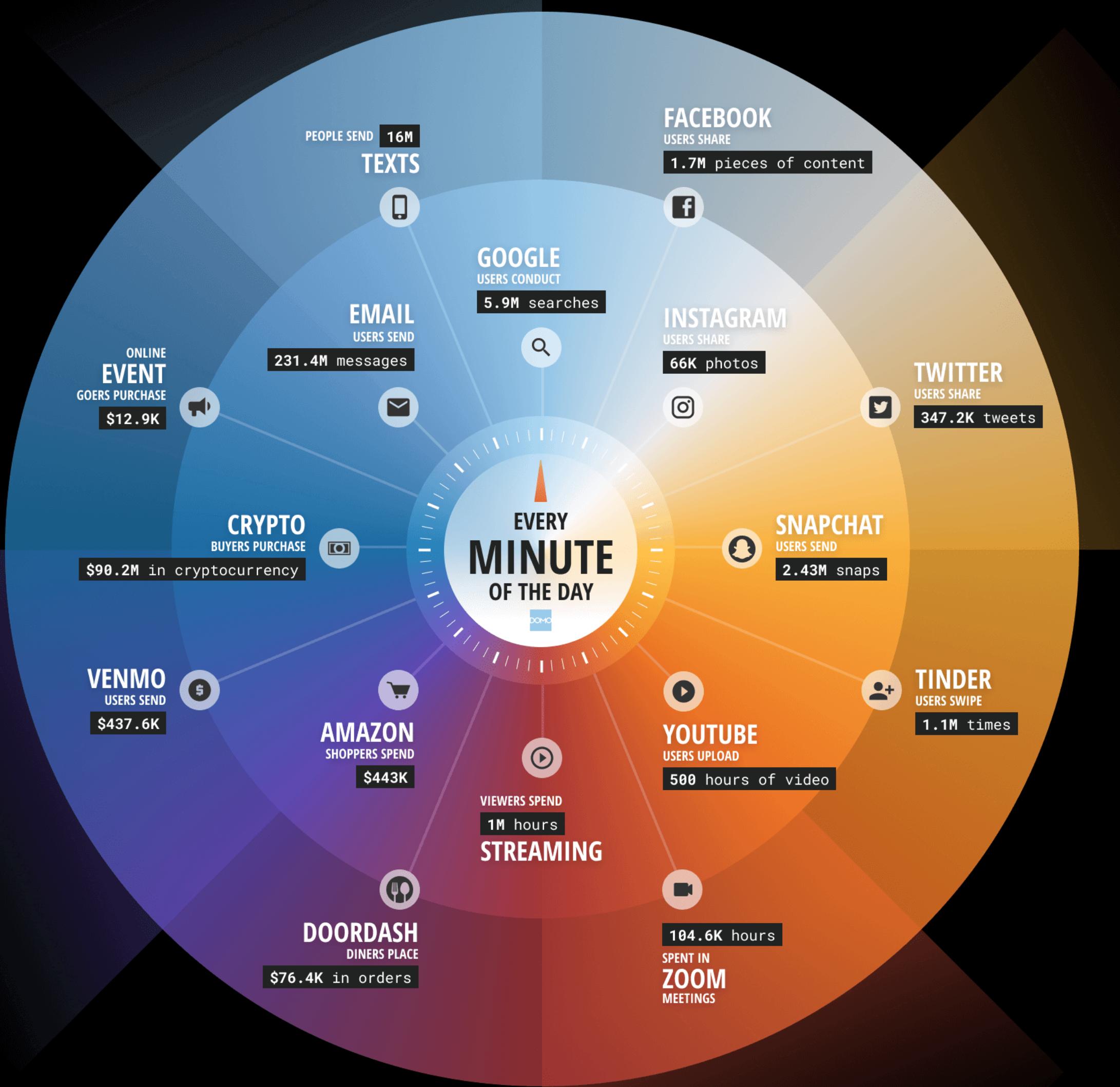
500M TB or 5 Exabytes



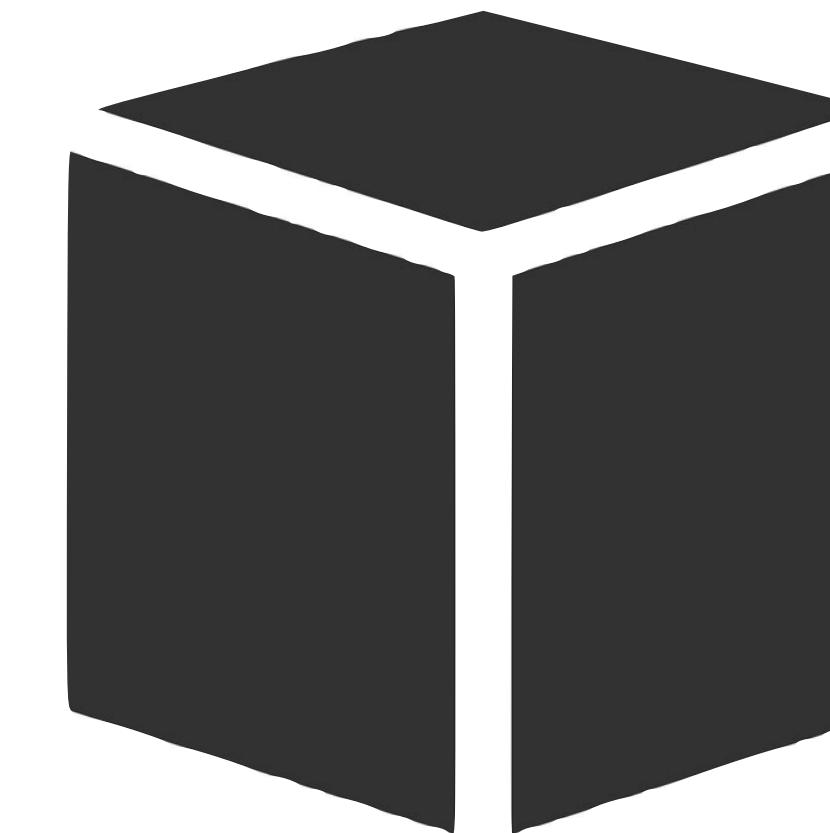
DATA NEVER SLEEPS 10.0

Over the last ten years, digital engagement through social media, streaming content, online purchasing, peer-to-peer payments and other activities has increased hundreds and even thousands of percentage points. While the world has faced a pandemic, economic ups and downs, and global unrest, there has been one constant in society:

our increasing use of new digital tools to support our personal and business needs, from connecting and communicating to conducting transactions and business. In this 10th annual "Data Never Sleeps" infographic, we share a glimpse at just how much data the internet produces each minute from some of this activity, marveling at the volume and variety of information that has been generated.



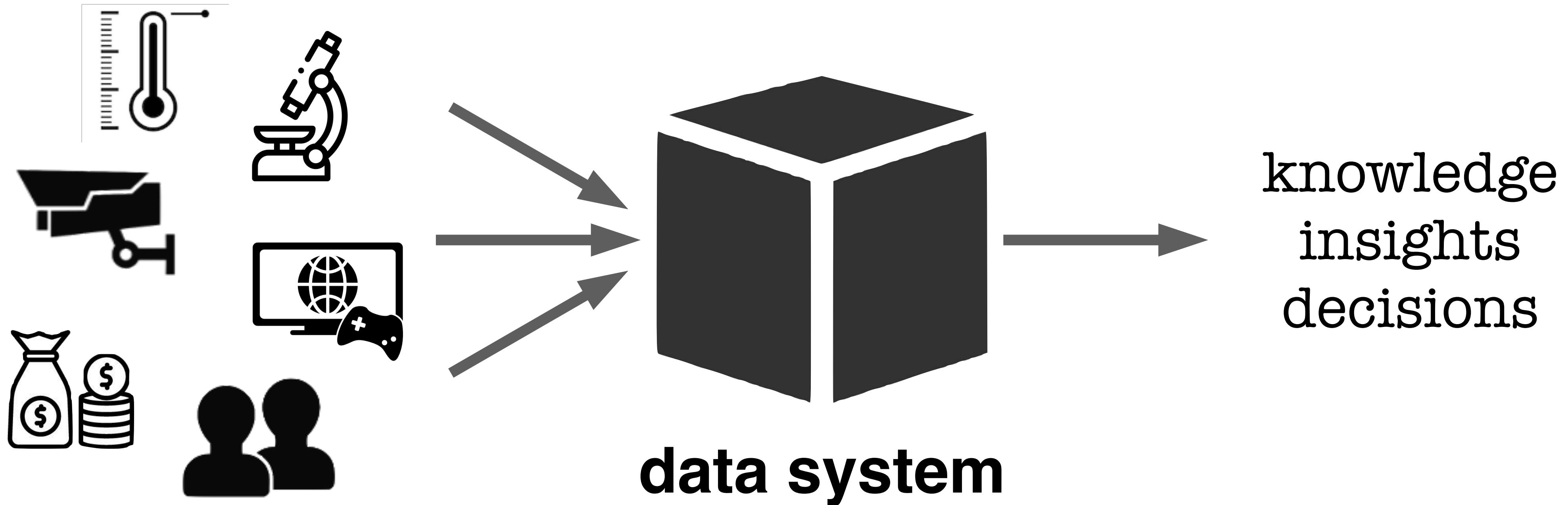
Managing big data
is a mammoth undertaking!
requires
specialized systems



data system /
data store /
DB kernel

Data systems

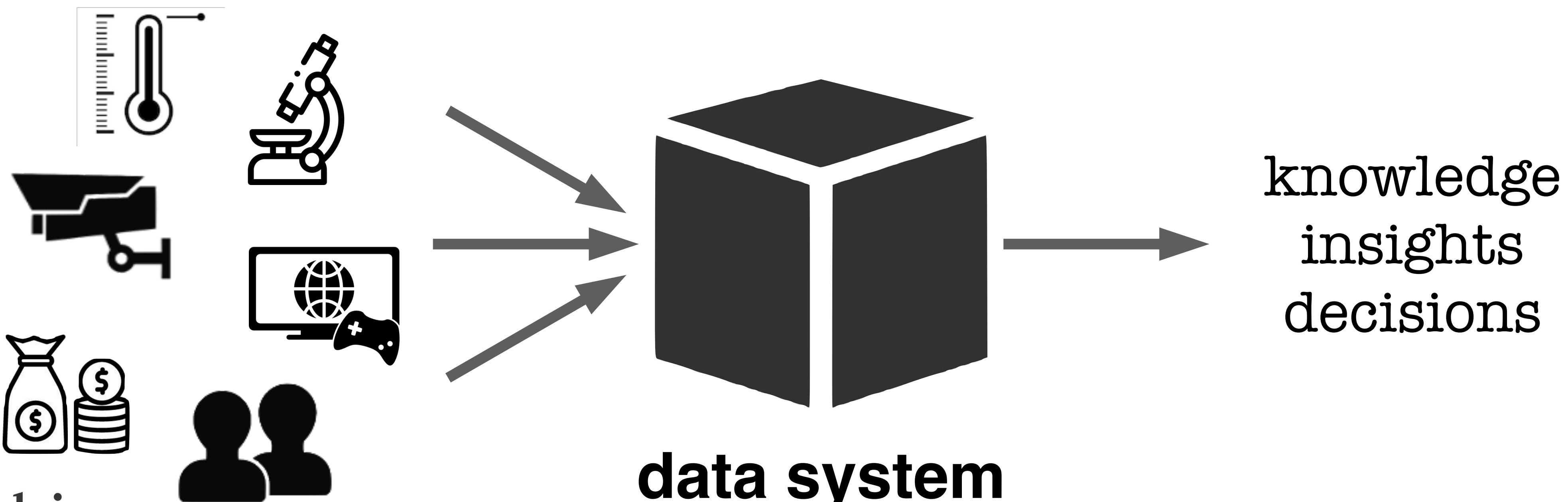
What are they, really?



Data systems

What are they, really?

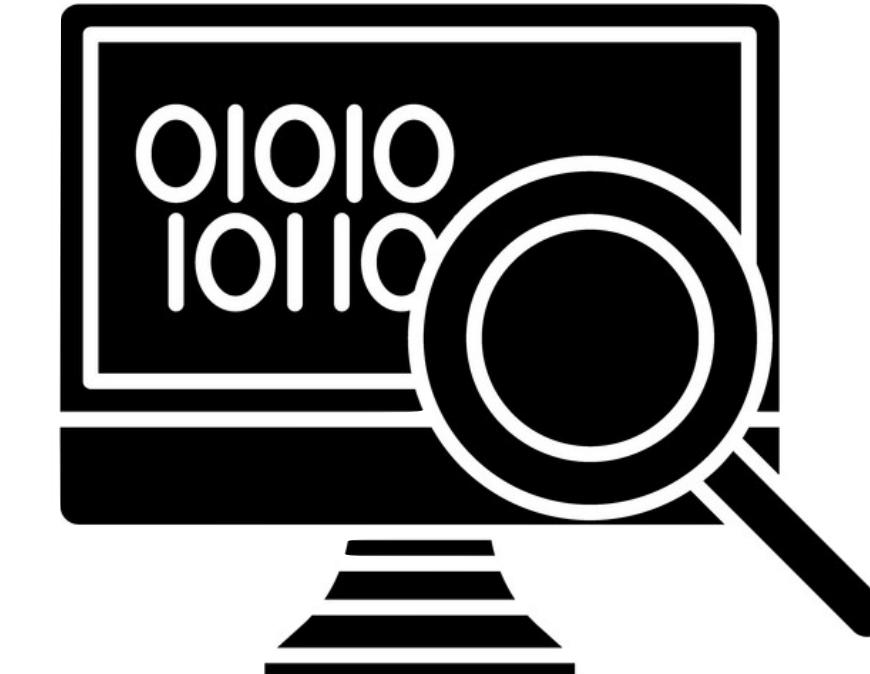
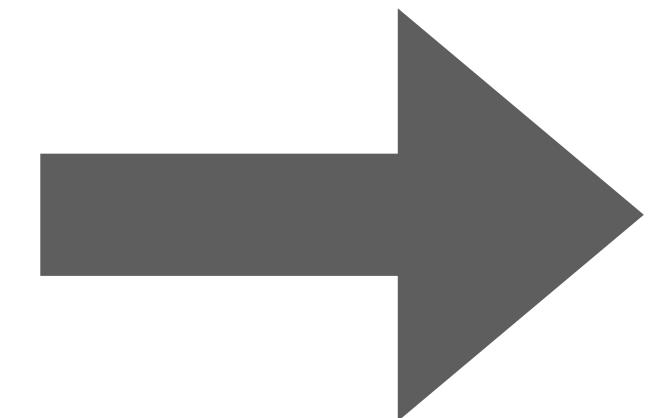
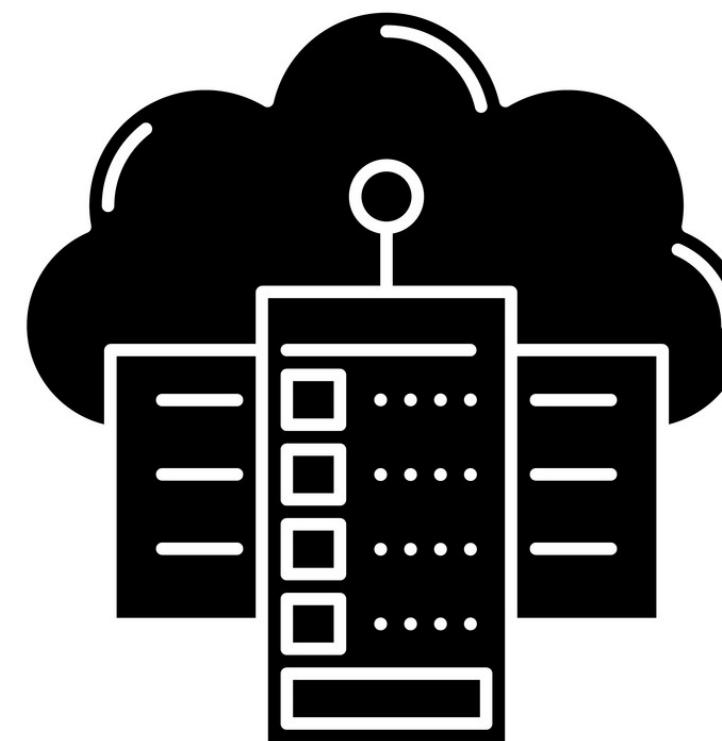
A data system is an **end-to-end software system** that is responsible for **storing data** and **providing access to the data** through **efficient data movement**.



Data systems

What are they, really?

A data system is an **end-to-end software system** that is responsible for **storing data** and **providing access to the data** through **efficient data movement**.

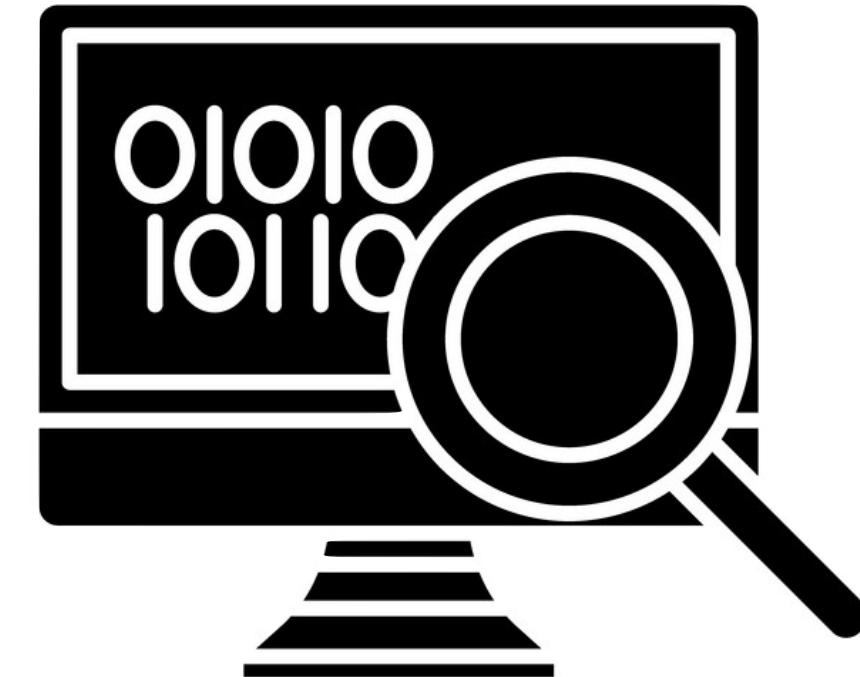
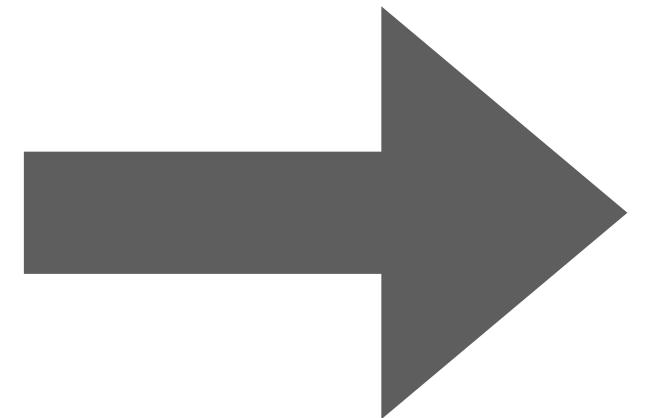
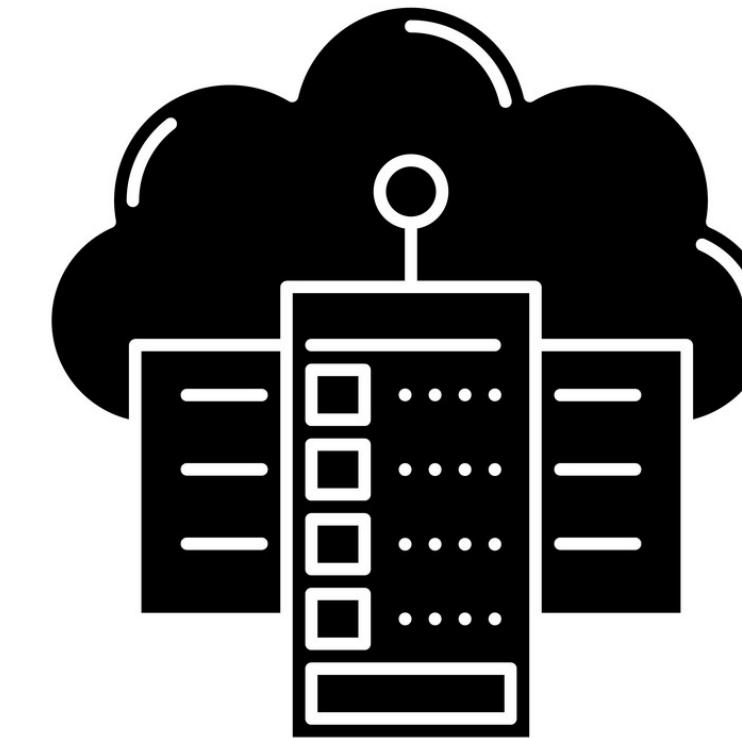


how we **store** data

how fast we can
access and **analyze** data

Data systems

What are they, really?



how we **store** data

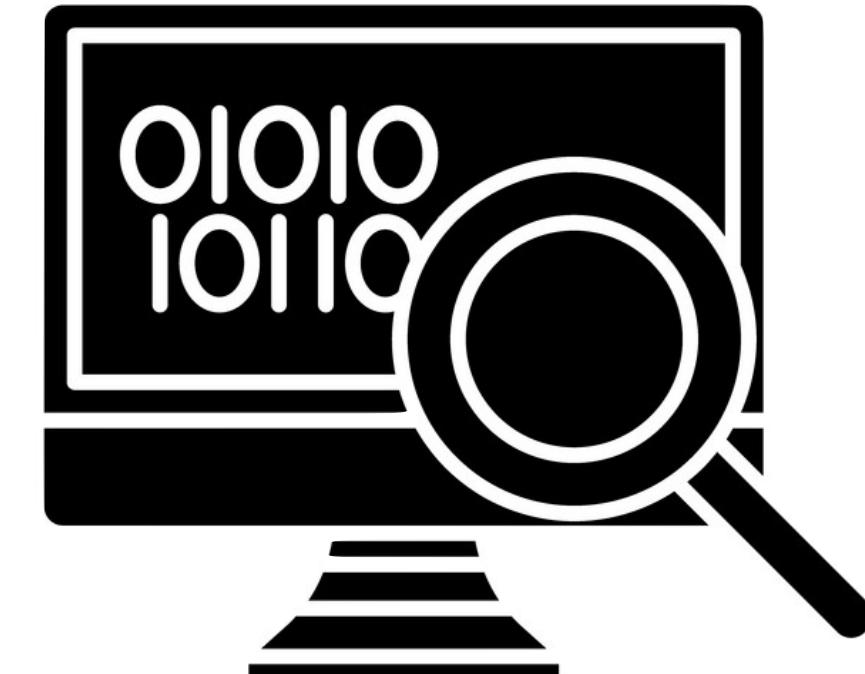
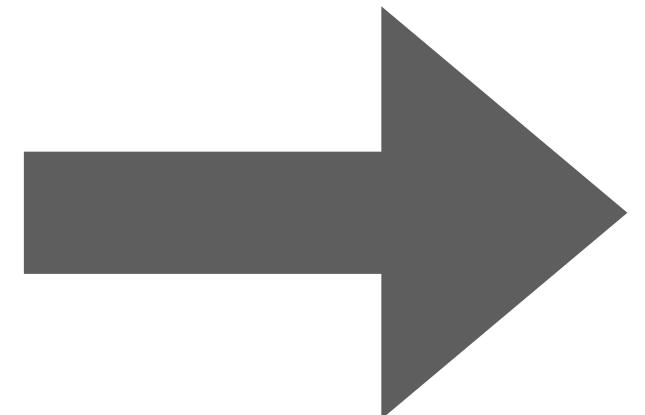
how fast we can
access and **analyze** data

70-80% of the **workload execution cost** comes from **data movement**

20-30% of it comes from in-memory processing

Data systems

What are they, really?



how we **store** data

how fast we can
access and **analyze** data

70-80% of the **workload execution cost** comes from **data movement**

20-30% of it comes from in-memory processing

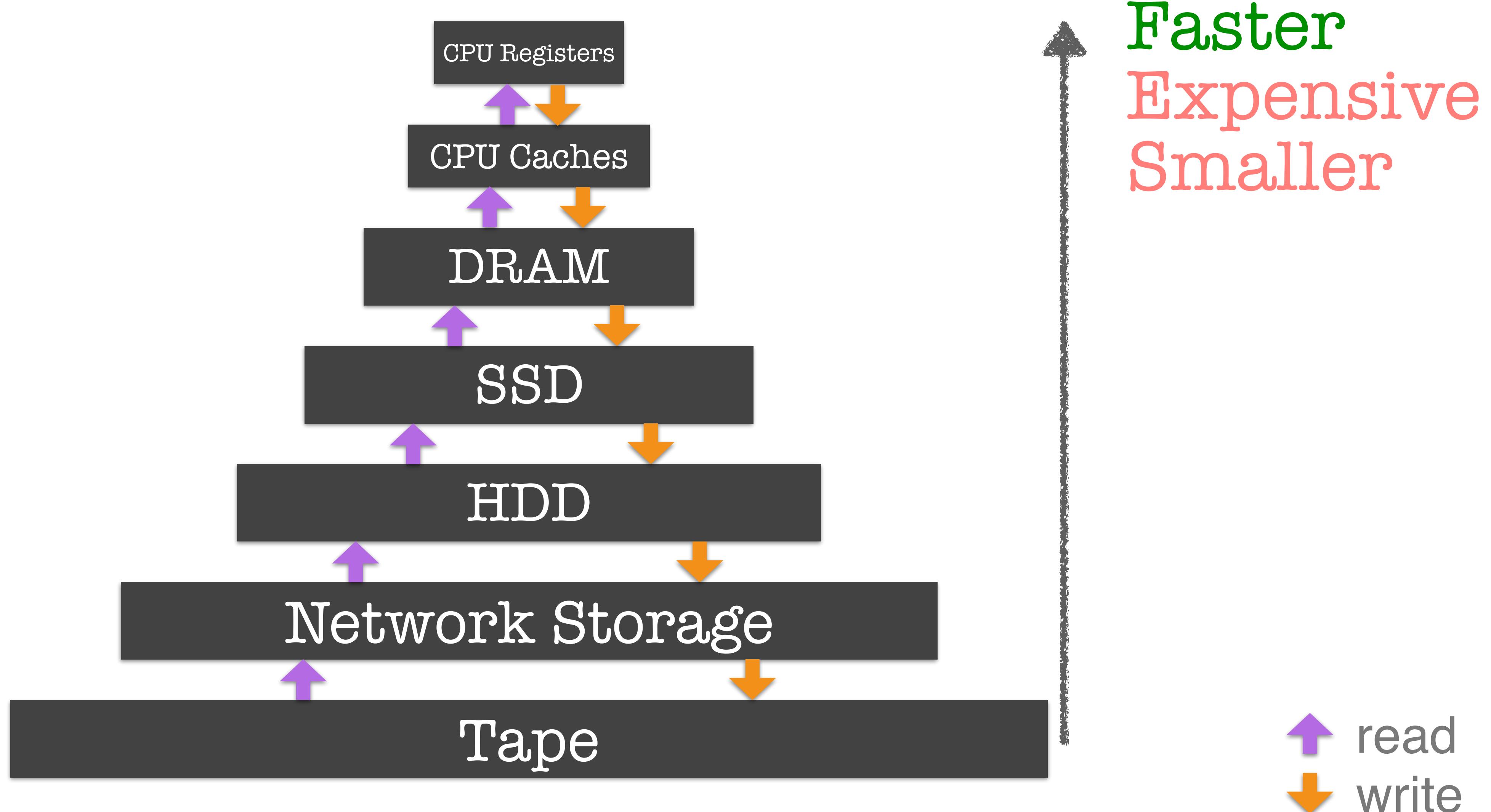
But why is **storage** such a **bottleneck**?



Storage hierarchy

Storage hierarchy

How data moves!

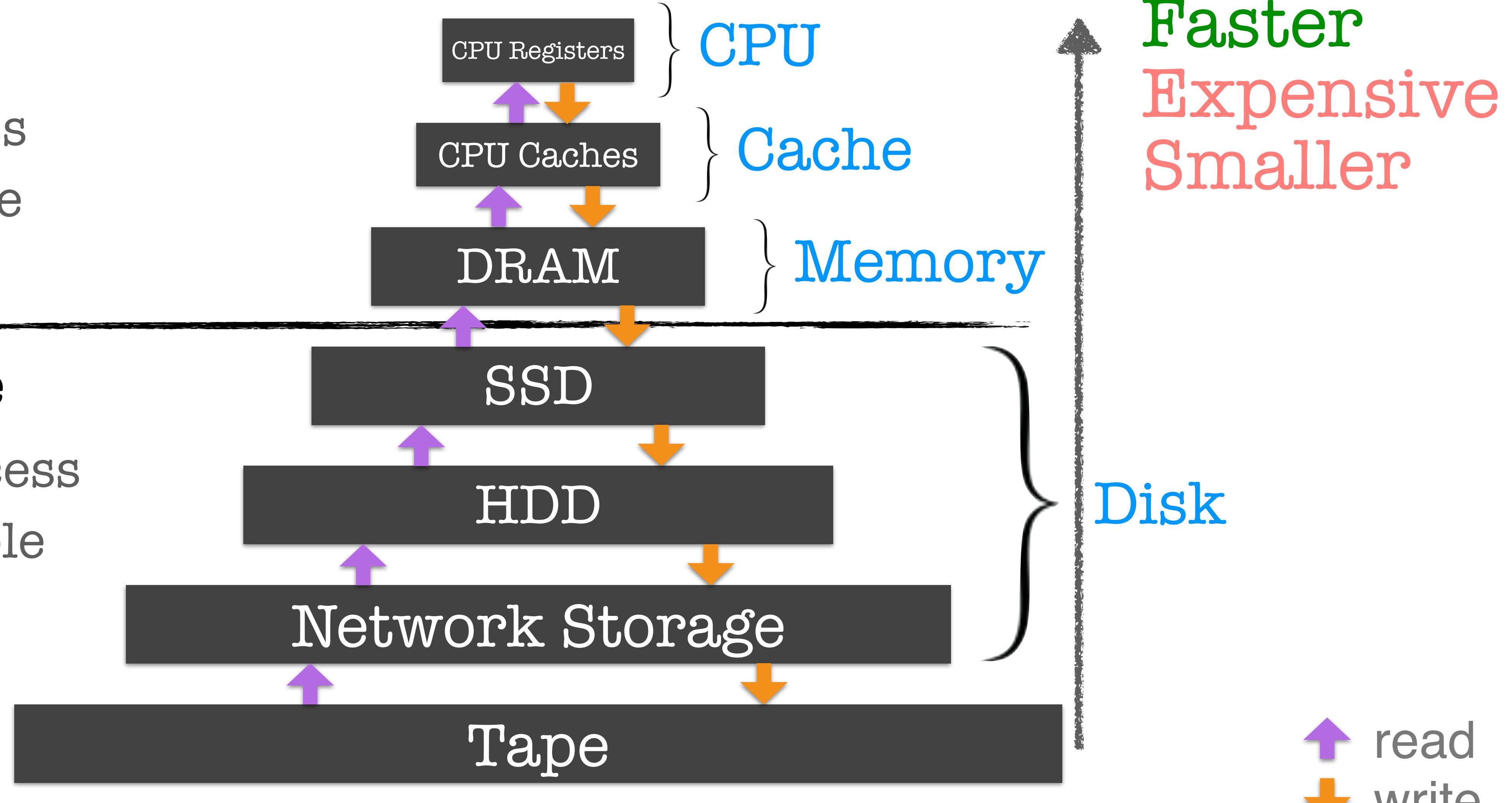


Storage hierarchy

How data moves!

Volatile
Random access
Byte accessible

Non-volatile
Sequential access
Block accessible



↑ read
↓ write



Brandeis
UNIVERSITY

Storage hierarchy

How data moves!

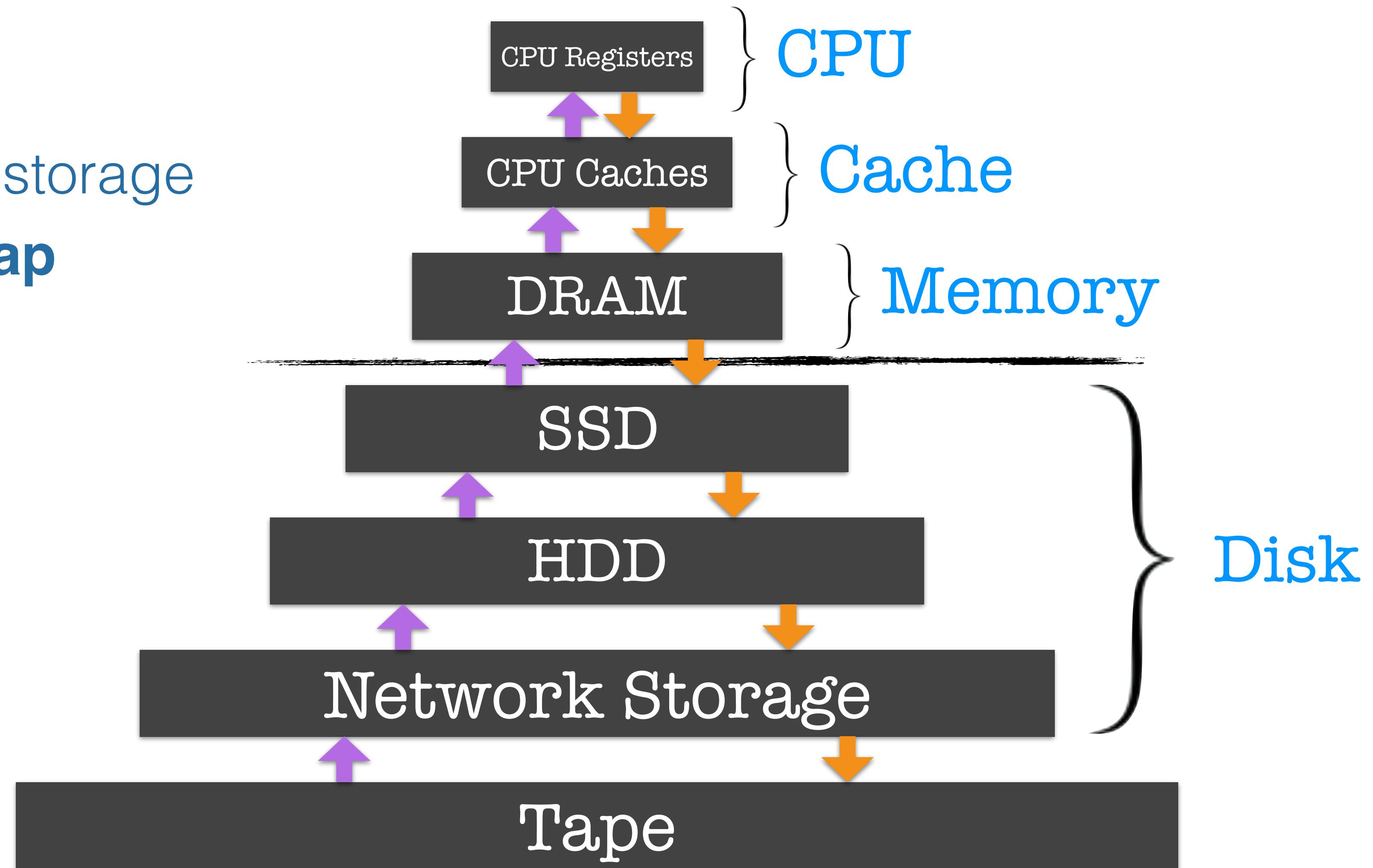
What is a **tape**?



sequential-only magnetic storage
super-slow but **super-cheap**
still a **multi-billion** industry



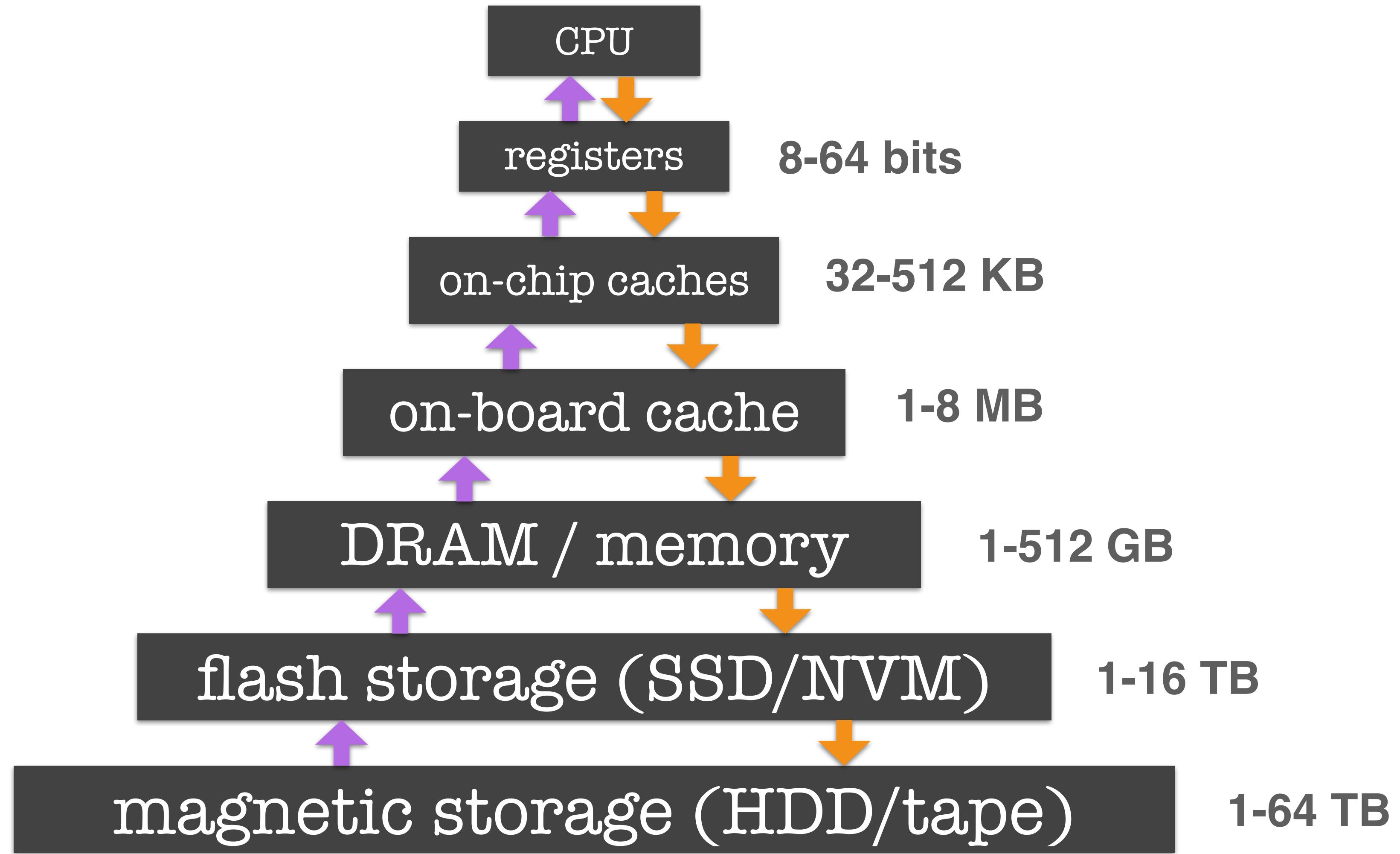
45TB @ \$150



Brandeis
UNIVERSITY

Storage hierarchy: updated

How data moves!



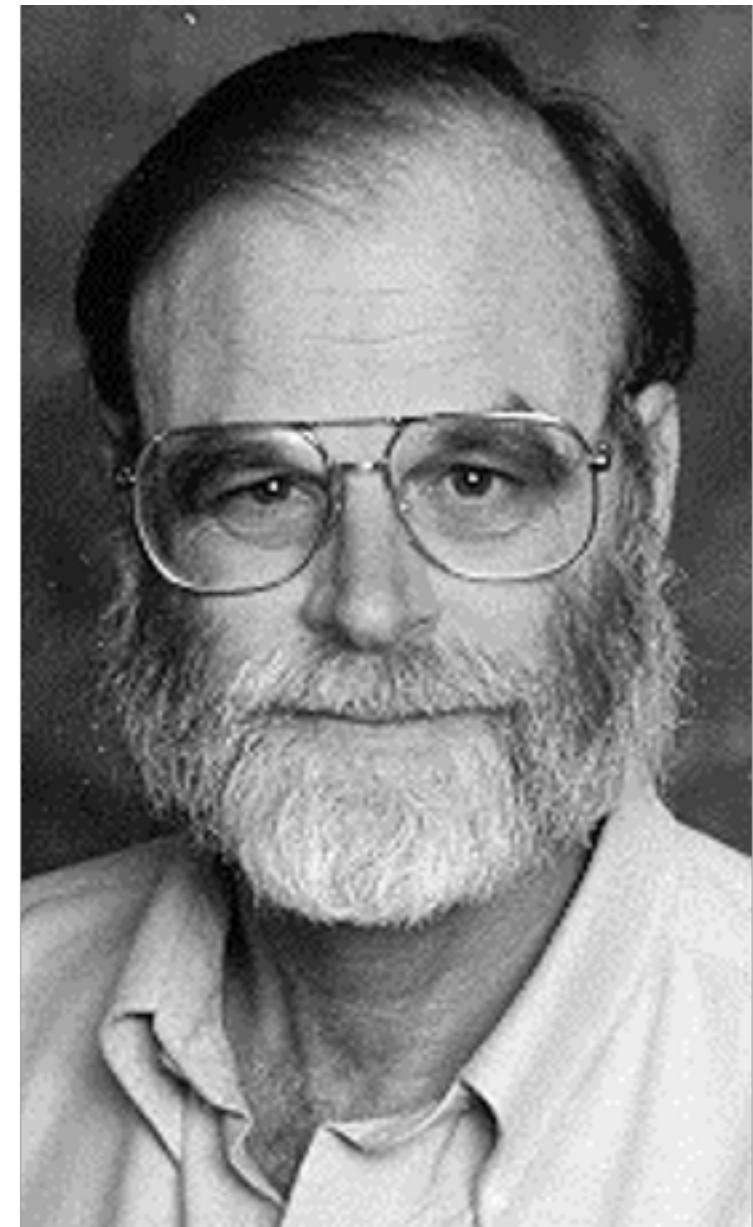
Latency numbers

How fast is access?

Access latency	Memory type
1 ns	CPU/register
4 ns	on-chip cache
100 ns	on-board cache
16,000 ns	DRAM
2,000,000 ns	SSD
50,000,000 ns	HDD
1,000,000,000 ns	Tape

Latency numbers

How fast is access?

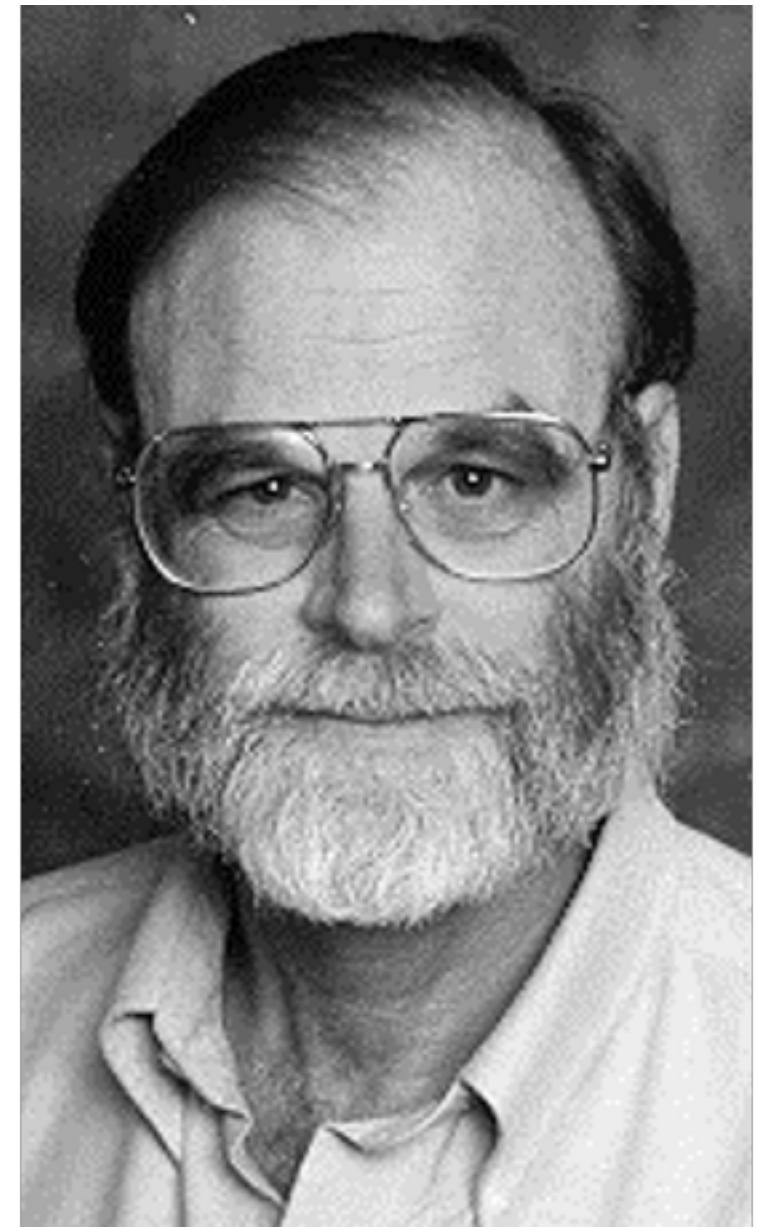


Jim Gray, IBM, Microsoft
ACM Turing Award 1998

Access latency	Memory type	Scaling up
1 ns	CPU/register	1 s
4 ns	on-chip cache	4 s
10 ns	on-board cache	10 s
100 ns	DRAM	100 s
16,000 ns	SSD	4.44 hours
2,000,000 ns	HDD	3.3 weeks
1,000,000,000 ns	Tape	31.7 years

Latency numbers

How fast is access?



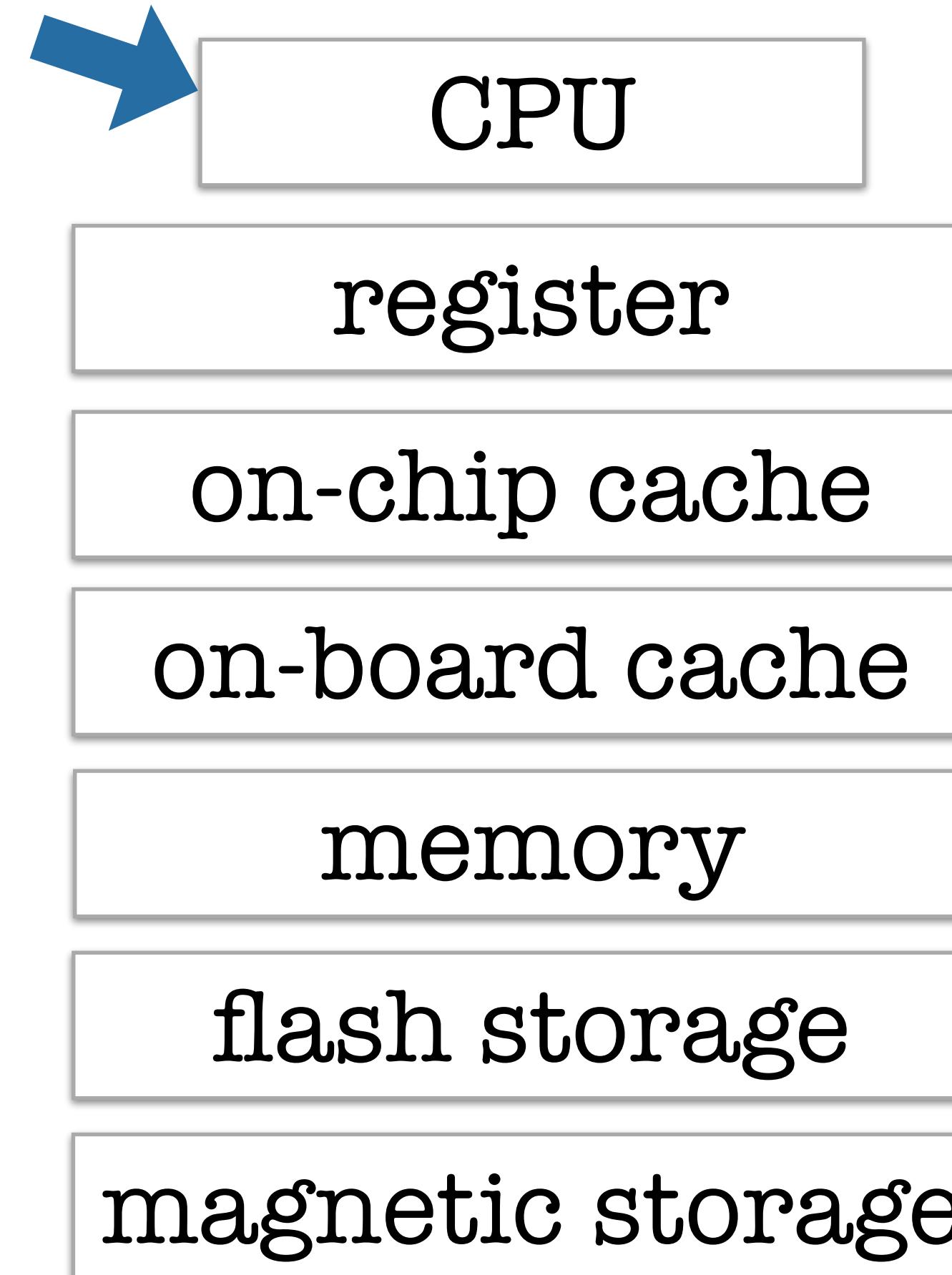
Jim Gray, IBM, Microsoft
ACM Turing Award 1998

Access latency	Memory type	Scaling up	Jim Gray's analogy
1 ns	CPU/register	1 s	My head
4 ns	on-chip cache	4 s	This room
10 ns	on-board cache	10 s	This building
100 ns	DRAM	100 s	Washington, DC
16,000 ns	SSD	4.44 hours	-
2,000,000 ns	HDD	3.3 weeks	Pluto
1,000,000,000 ns	Tape	31.7 years	Andromeda

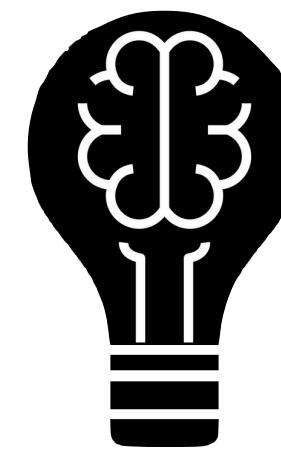
Disk (secondary storage) is **SLOW!**

Memory wall

computations
happen here



Try not to jump the wall



Thought Experiment 1

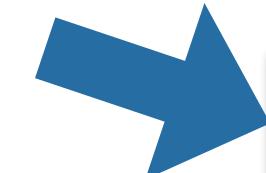
What if data is **not found in cache?**

cache miss!

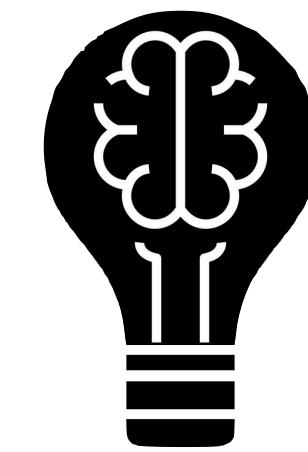
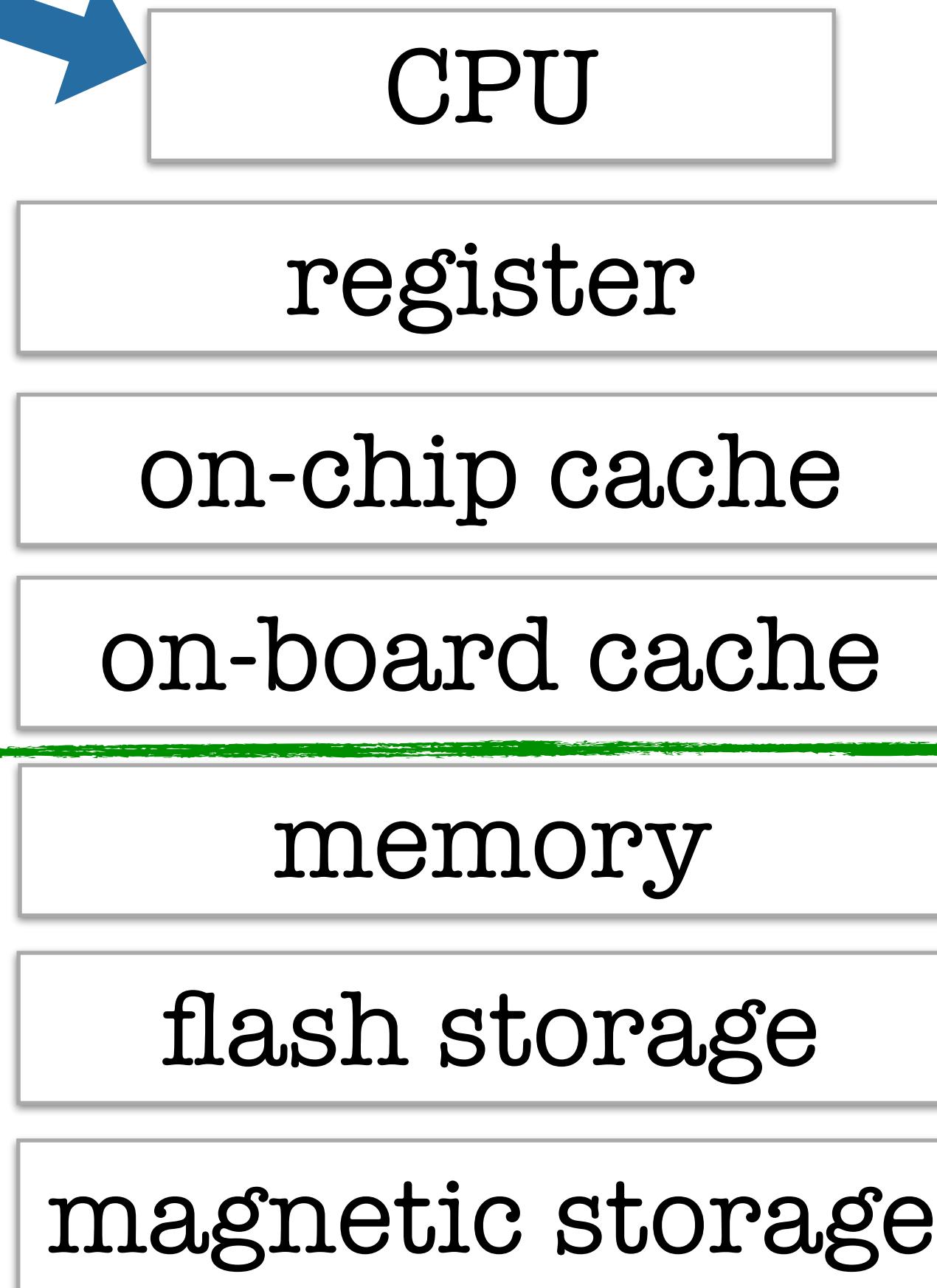


Memory wall

computations
happen here



Try not to jump the wall

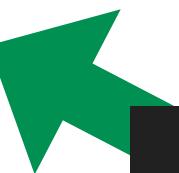


Thought Experiment 1

What if data is **not found in cache?**

cache miss

looking for something
that is **not in the cache**



be **careful** when you go below the **green line**

Memory wall

computations
happen here



CPU

Try not to jump the wall

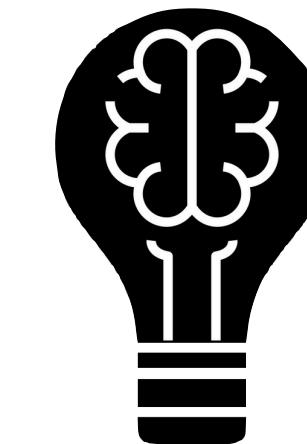
be **careful** when you go below the **green line**

register

on-chip cache

10 ns

on-board cache



Thought Experiment 2

100 ns

memory

What if data is **not found in memory?**
memory miss!

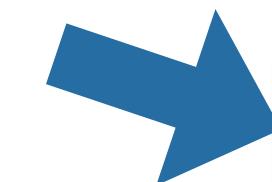
flash storage

magnetic storage



Memory wall

computations
happen here



CPU

Try not to jump the wall

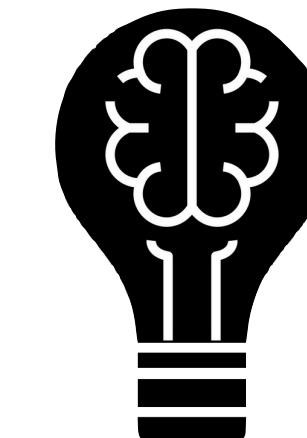
be **careful** when you go below the **green line**

register

on-chip cache

10 ns

on-board cache



Thought Experiment 2

What if data is **not found in memory?**

100 ns

memory

memory miss

looking for something that
is **not in the memory**

16,000 ns

flash storage

2 ms

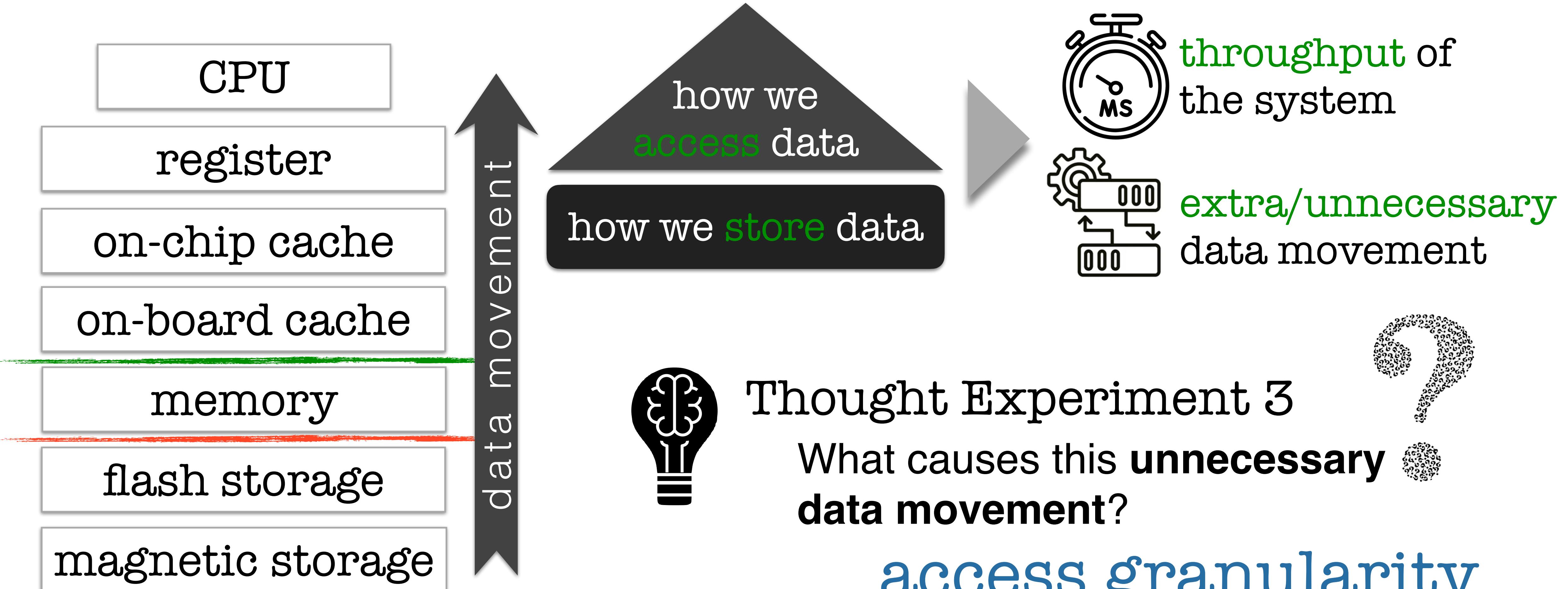
magnetic storage

be **VERY careful** when you go below the **red line**



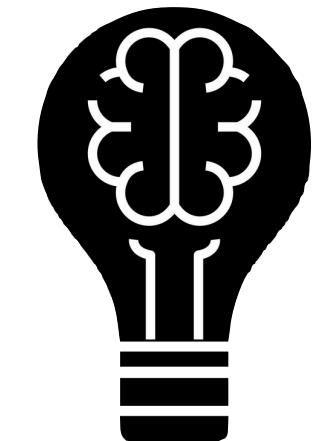
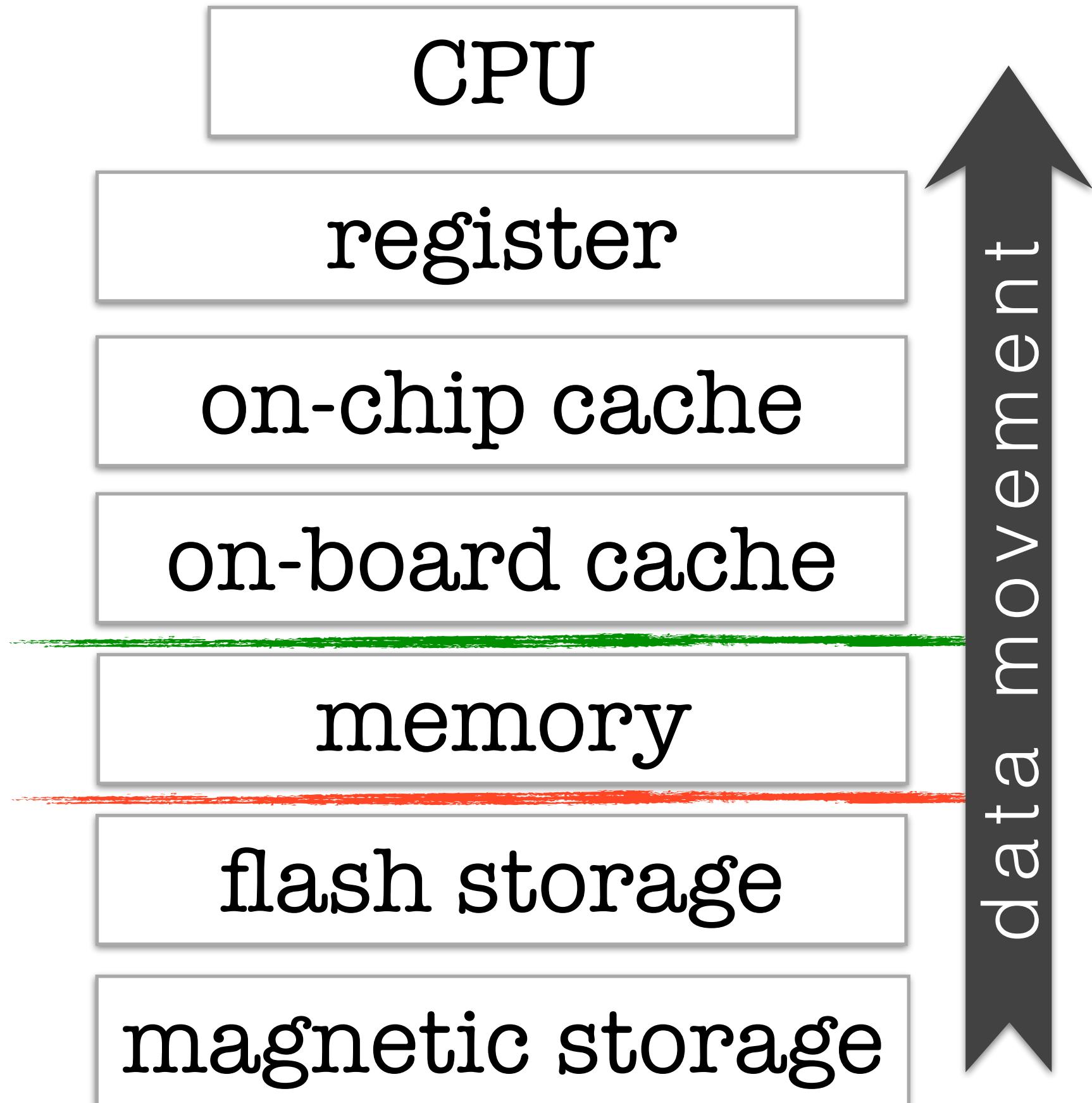
Access granularity

How much data to move



Access granularity

How much data to move

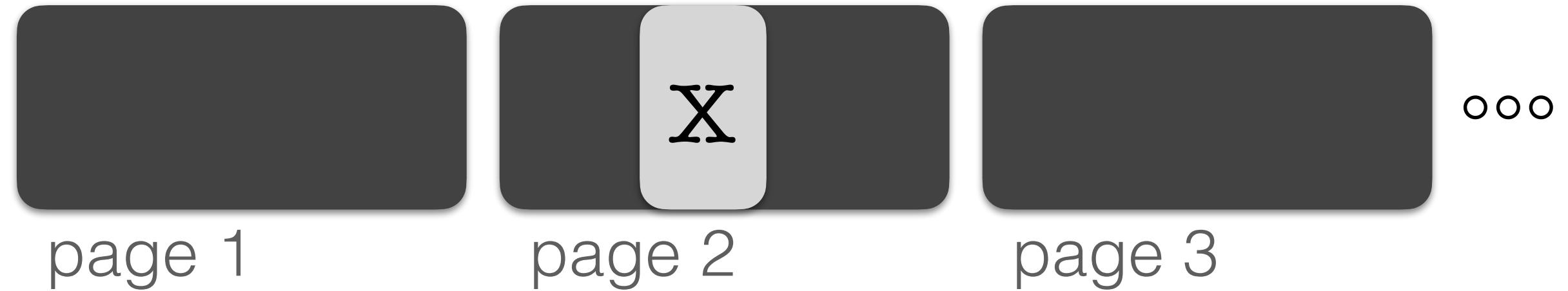


Thought Experiment 3

What causes this **unnecessary data movement?**

access granularity

get(X)

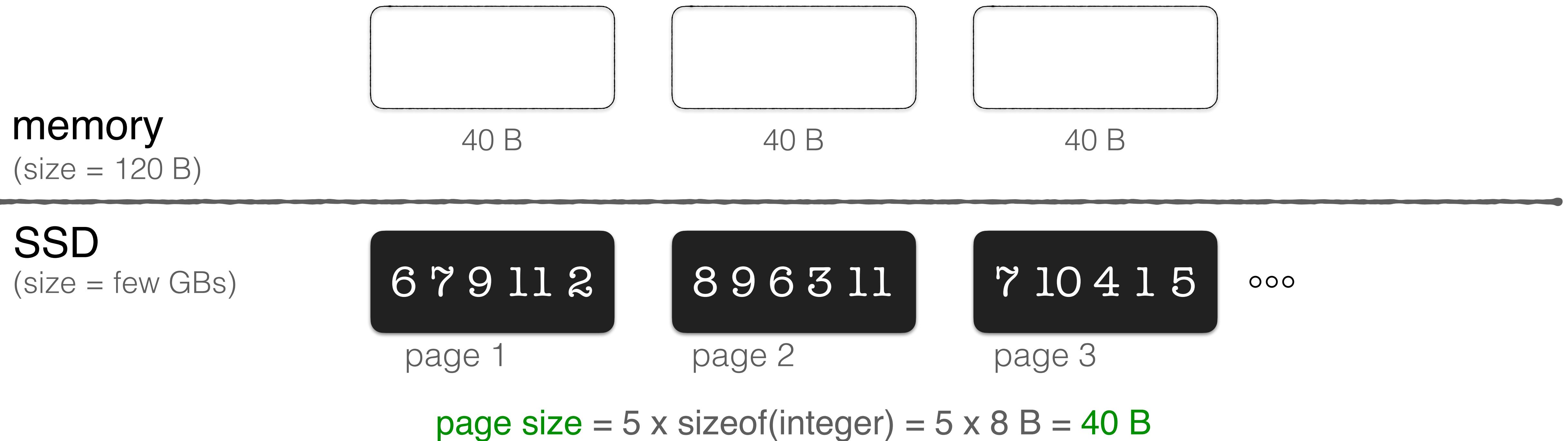


we **cannot** read only X
have to read the **whole page**
this comes from the **hardware**



Let's think together

Understanding the implications of access granularity



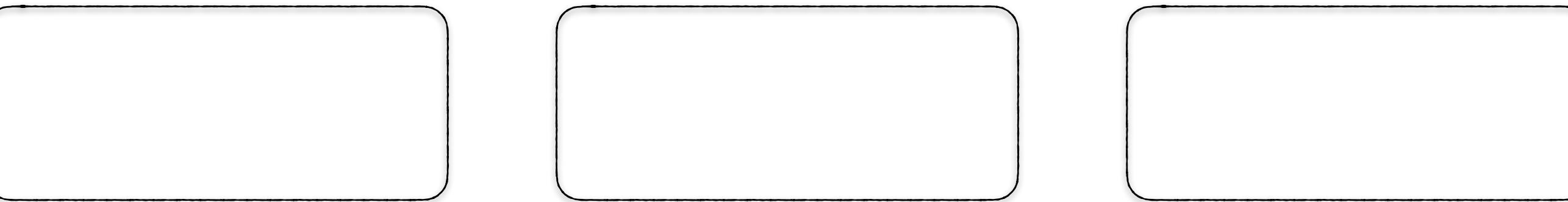
Let's think together

Understanding the implications of access granularity

query: $x < 4$

memory
(size = 120 B)

SSD
(size = few GBs)



6 7 9 11 2

page 1

8 9 6 3 11

page 2

7 10 4 1 5

page 3

ooo

page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

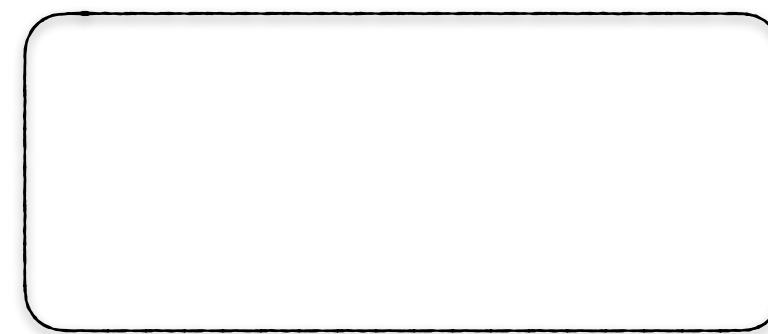
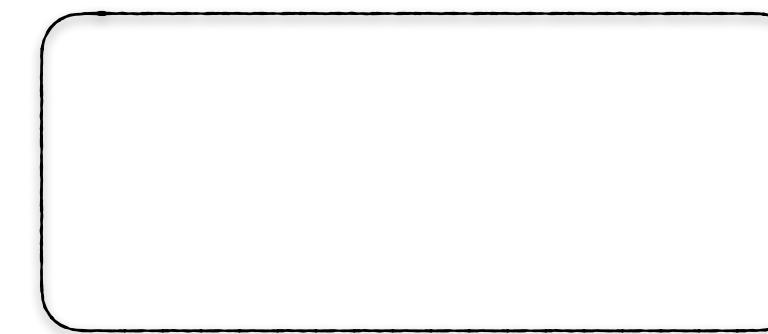
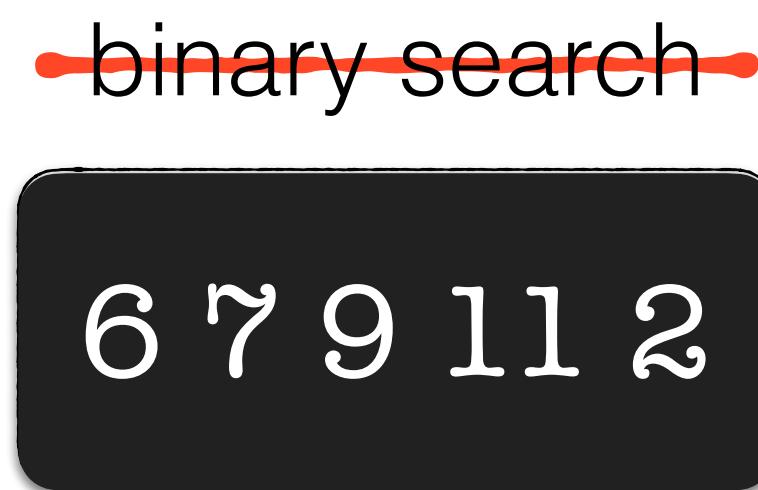
Let's think together

Understanding the implications of access granularity

query: $x < 4$

memory
(size = 120 B)

SSD
(size = few GBs)



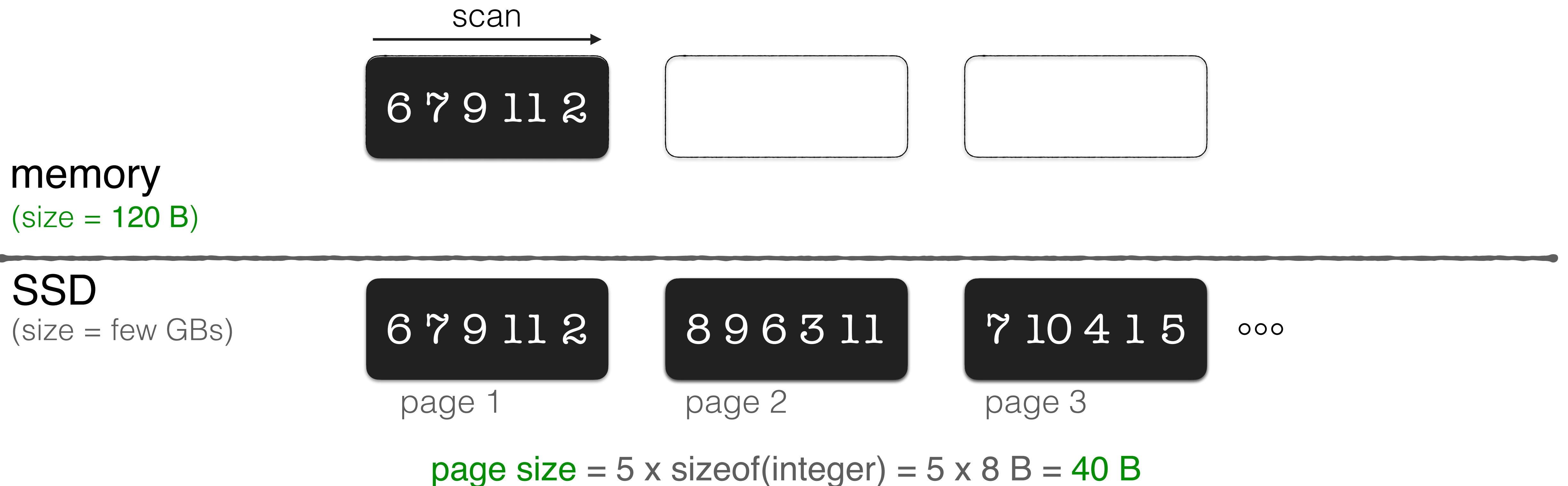
ooo

page size = $5 \times \text{sizeof(integer)} = 5 \times 8 \text{ B} = 40 \text{ B}$

Let's think together

Understanding the implications of access granularity

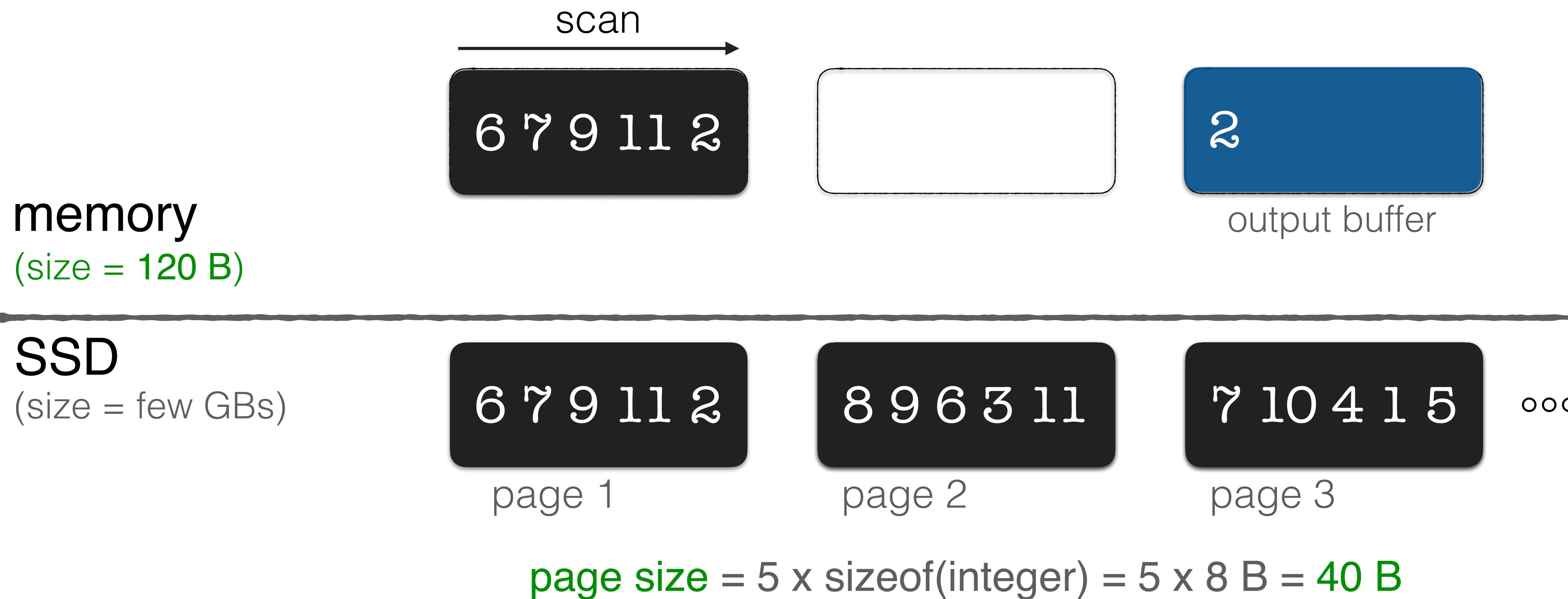
query: $x < 4$



Let's think together

Understanding the implications of access granularity

query: $x < 4$



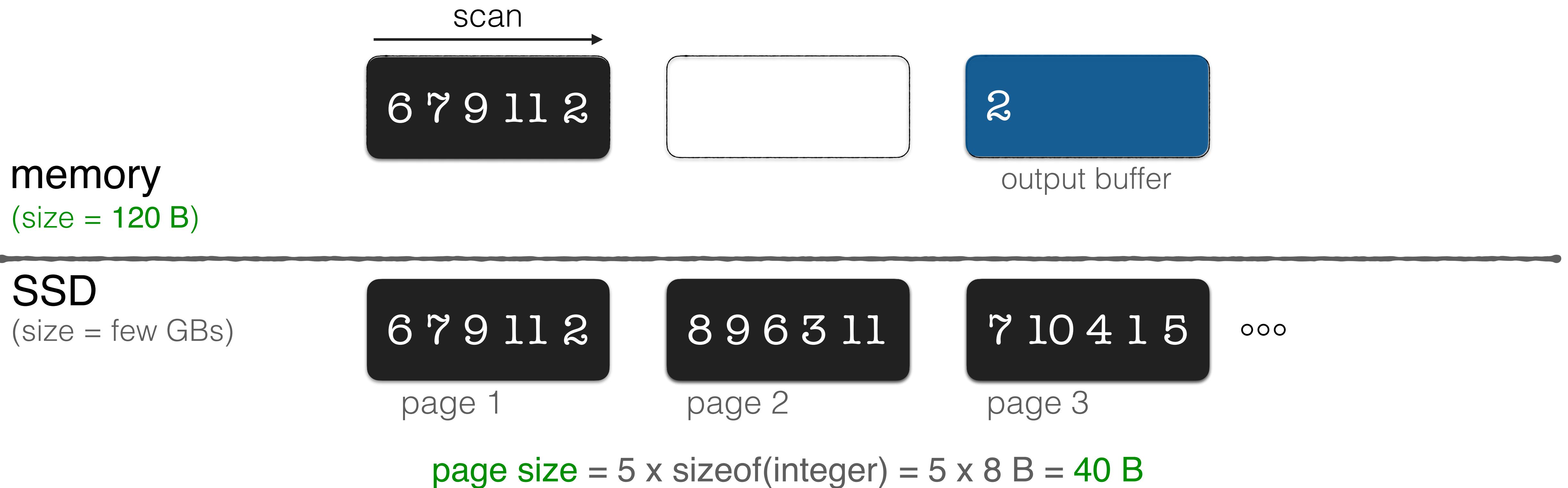
Let's think together

Understanding the implications of access granularity

query: $x < 4$



40 B (1 I/O)



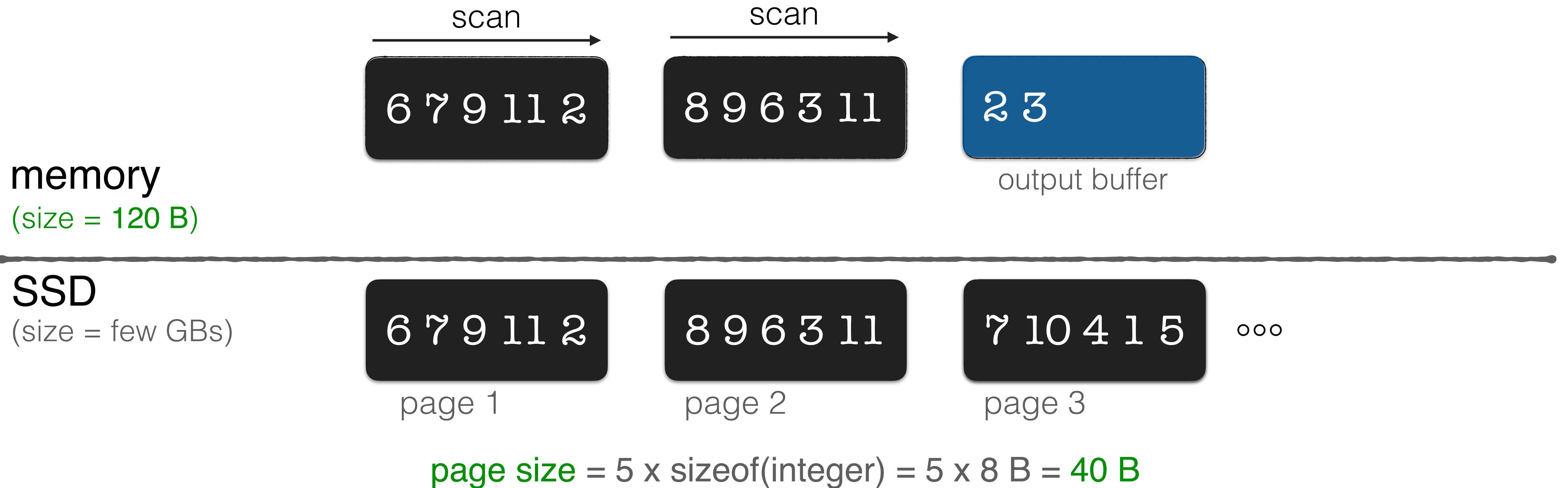
Let's think together

Understanding the implications of access granularity

query: $x < 4$



40 B (1 I/O)



Let's think together

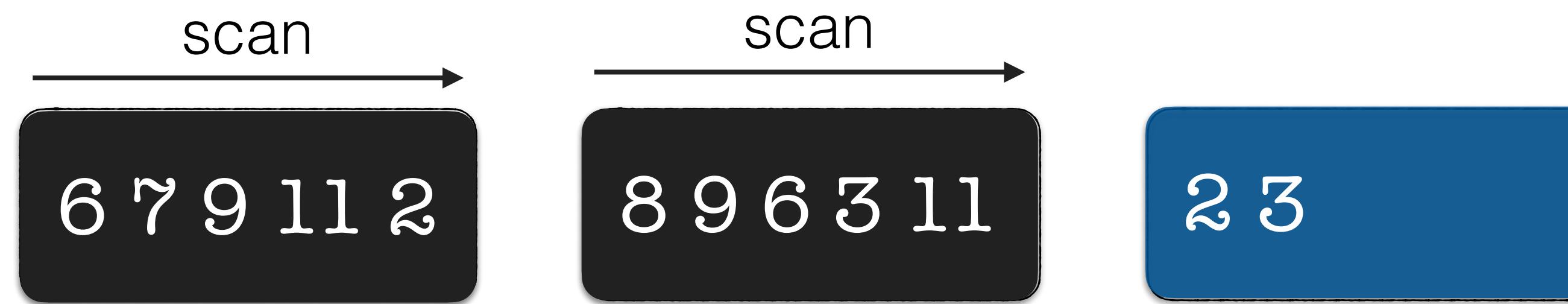
Understanding the implications of access granularity

query: $x < 4$

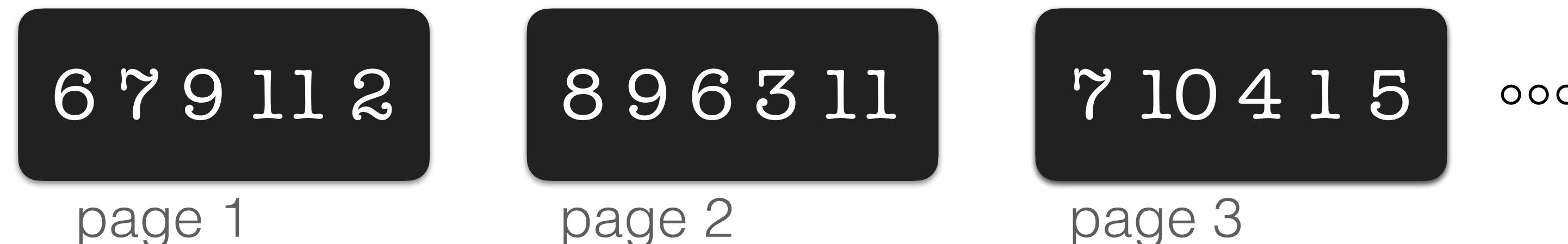


80 B (2 I/O)

memory
(size = 120 B)



SSD
(size = few GBs)



page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

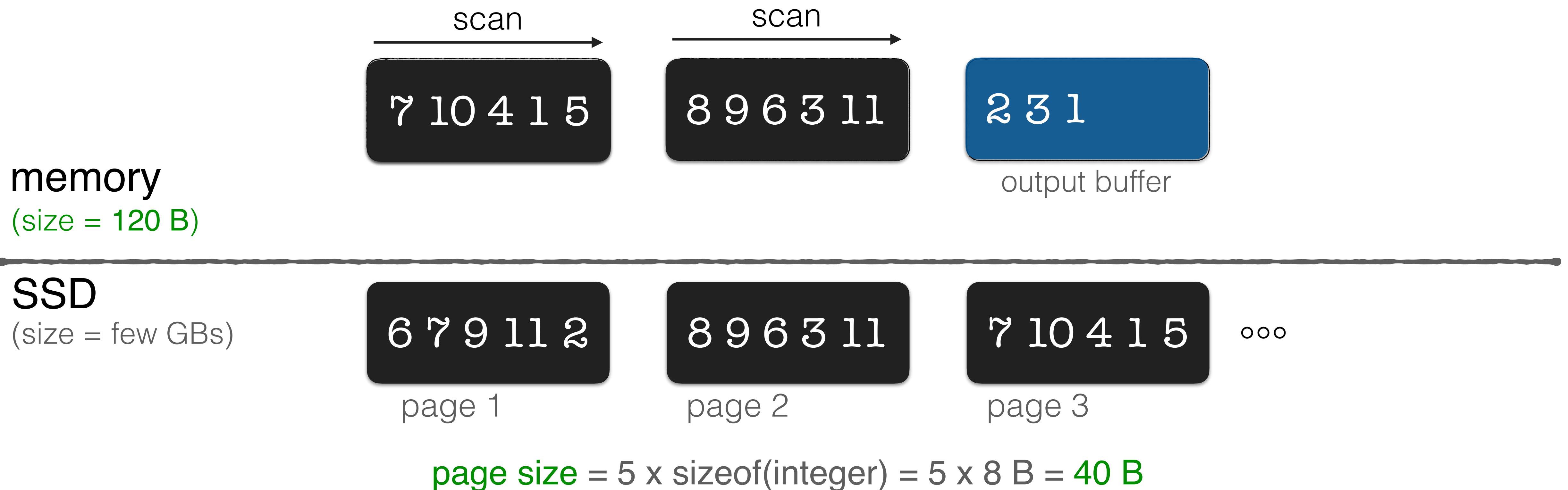
Let's think together

Understanding the implications of access granularity

query: $x < 4$

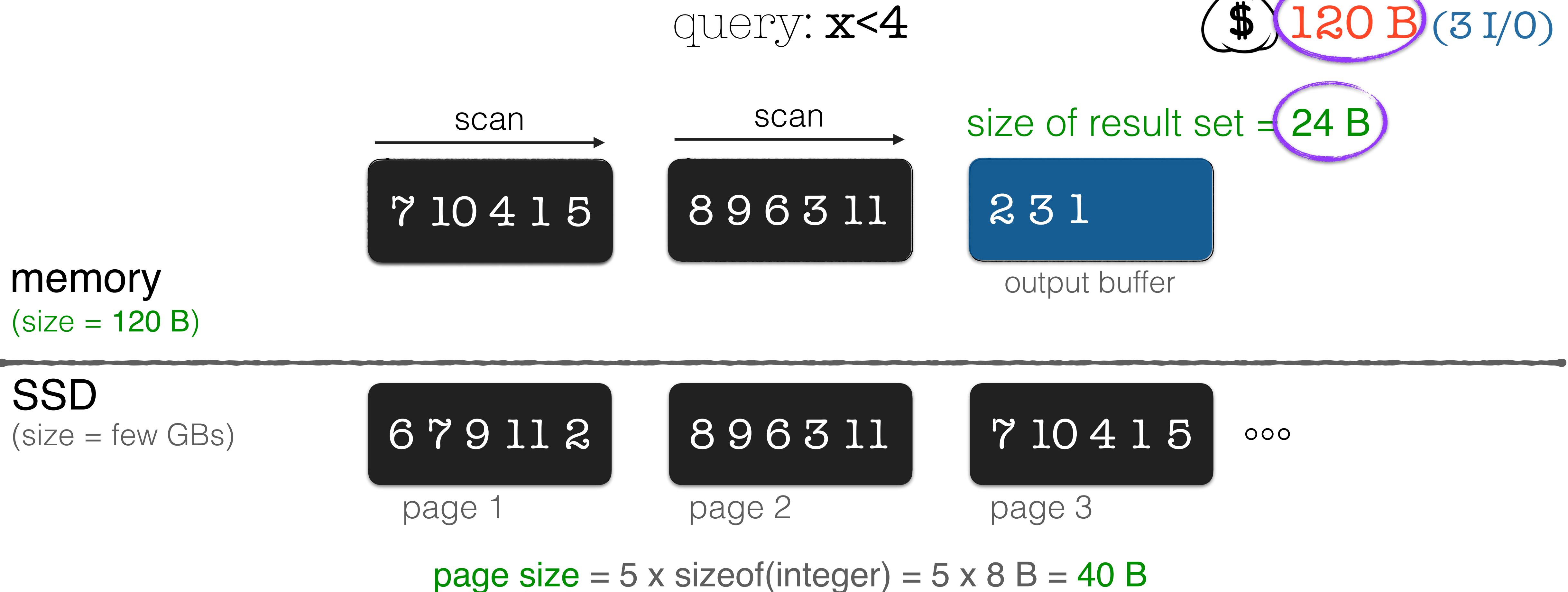


80 B (2 I/O)



Let's think together

Understanding the implications of access granularity

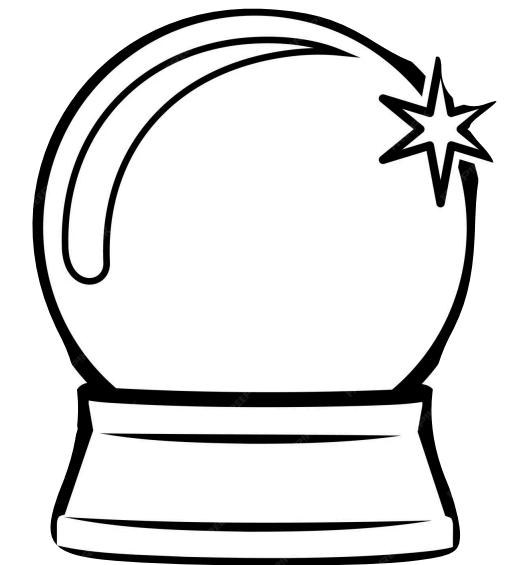


What if we had an **oracle**?



What if we had the **perfect index**?



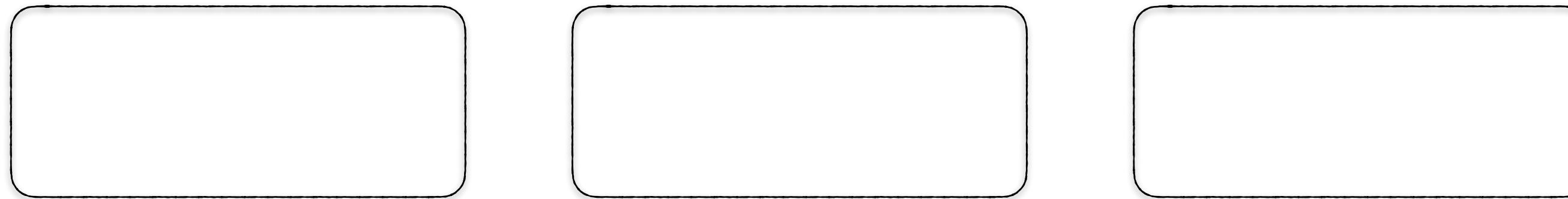


Let's think together

Understanding the implications of access granularity

query: $x < 4$

gives us the **exact position**
of the **qualifying entries**



memory
(size = 120 B)

SSD
(size = few GBs)

6 7 9 11 2

page 1

8 9 6 3 11

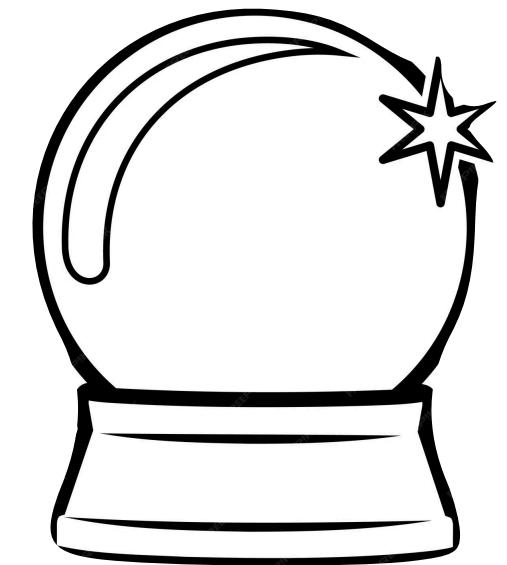
page 2

7 10 4 1 5

page 3

ooo

page size = 5 x sizeof(integer) = 5 x 8 B = 40 B



Let's think together

Understanding the implications of access granularity

query: $x < 4$

gives us the **exact position**
of the **qualifying entries**

memory
(size = 120 B)

SSD
(size = few GBs)

6 7 9 11 2



6 7 9 11 2

8 9 6 3 11

7 10 4 1 5

ooo

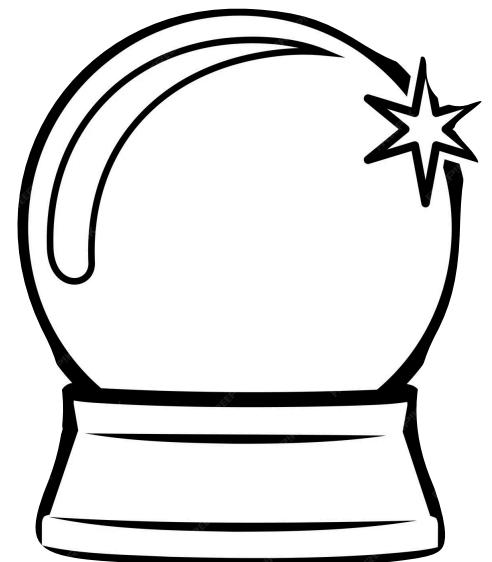
page 1

page 2

page 3

page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

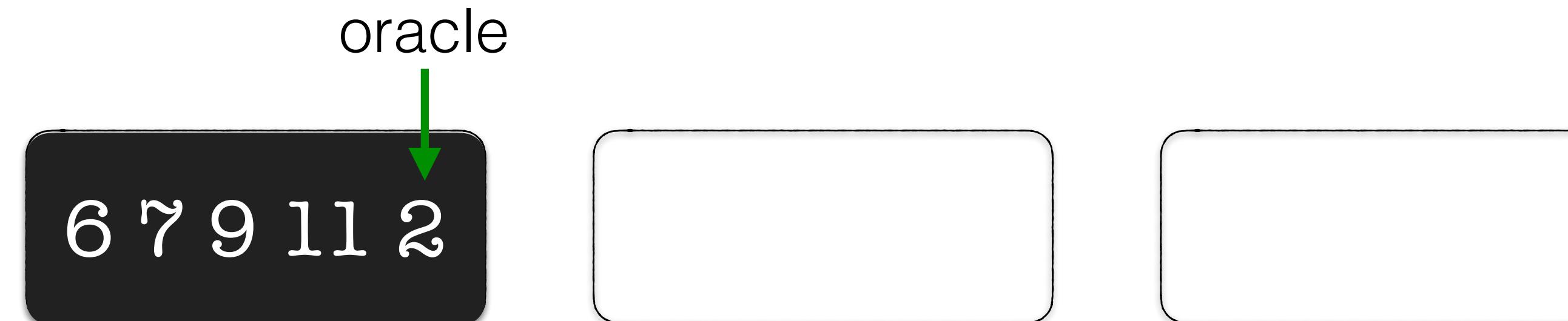
Let's think together



gives us the **exact position** of the **qualifying entries**

memory
(size = 120 B)

SSD
(size = few GBs)



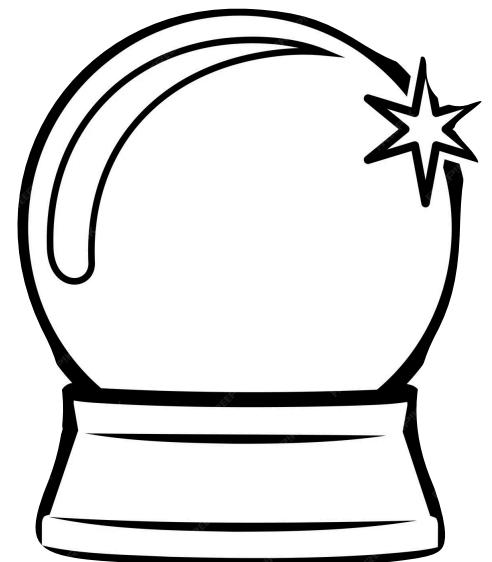
page 1

page 2

page 3

page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

Let's think together



Understanding the implications of access granularity

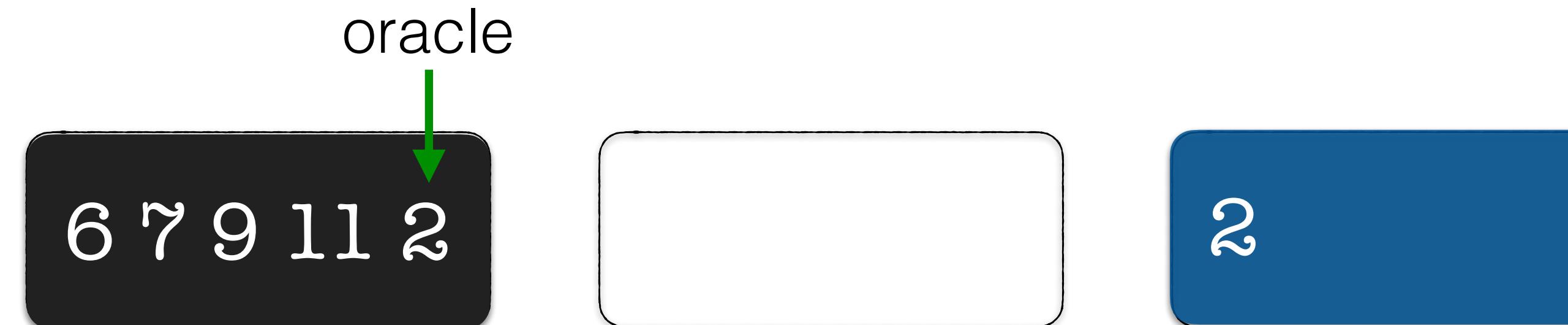


query: $x < 4$

gives us the **exact position**
of the **qualifying entries**

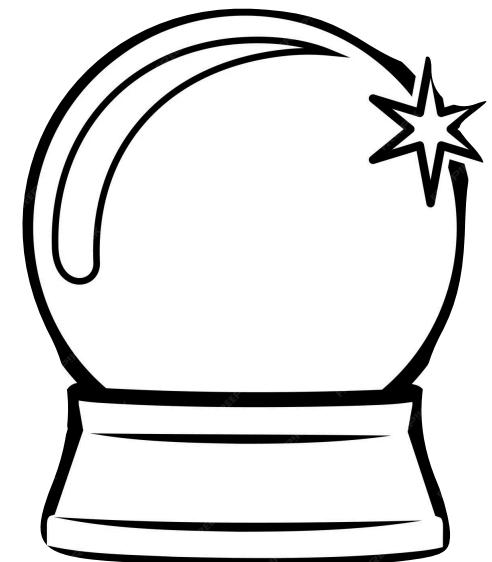
memory
(size = 120 B)

SSD
(size = few GBs)



page size = $5 \times \text{sizeof(integer)} = 5 \times 8 \text{ B} = 40 \text{ B}$

Let's think together



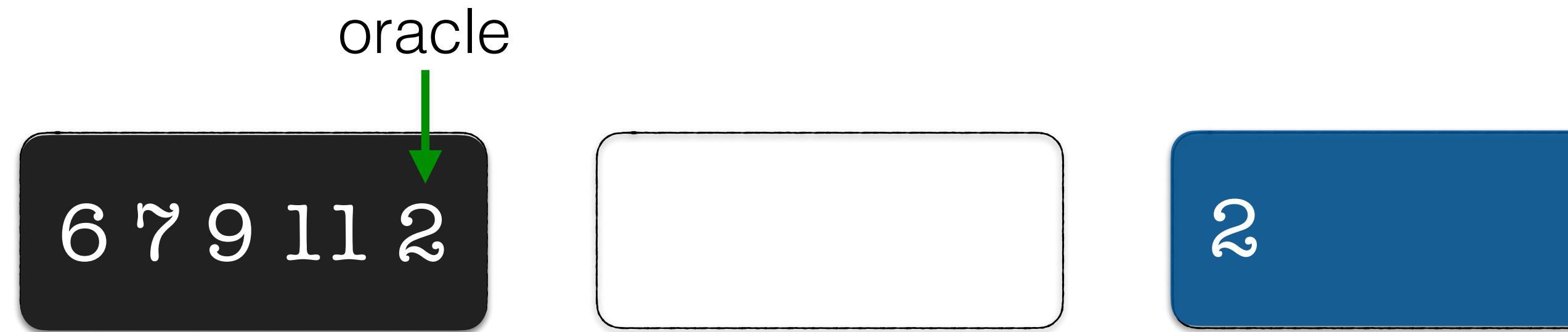
Understanding the implications of access granularity



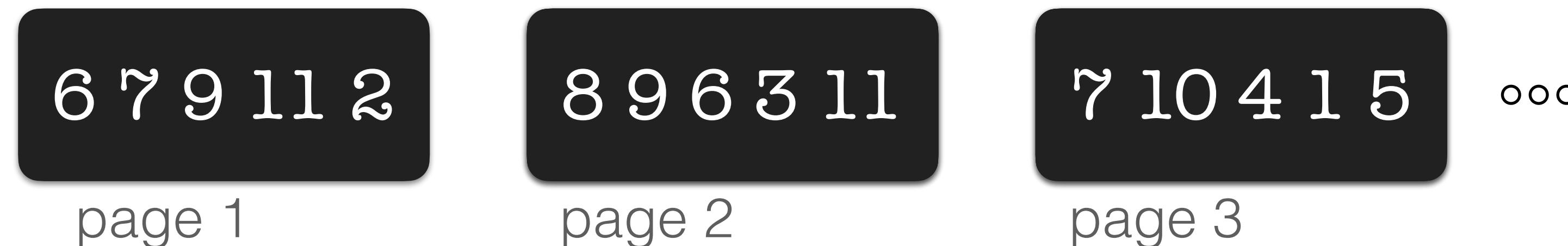
40 B (1 I/O)

gives us the **exact position**
of the **qualifying entries**

memory
(size = 120 B)

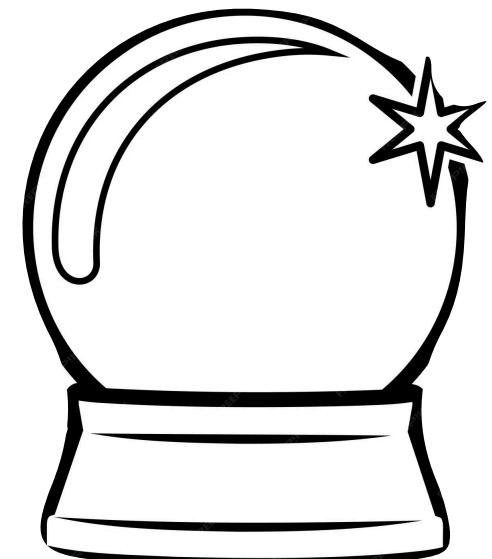


SSD
(size = few GBs)



page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

Let's think together



Understanding the implications of access granularity

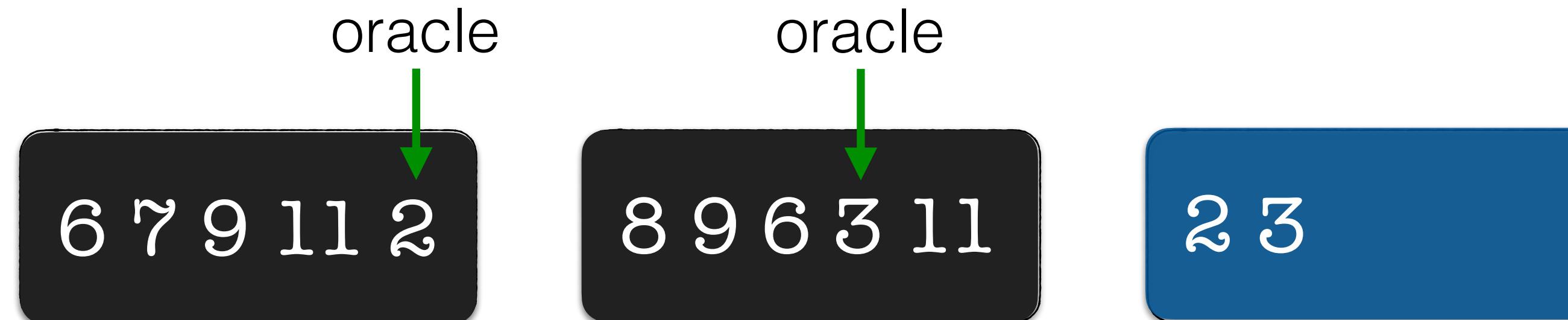


40 B (1 I/O)

gives us the **exact position**
of the **qualifying entries**

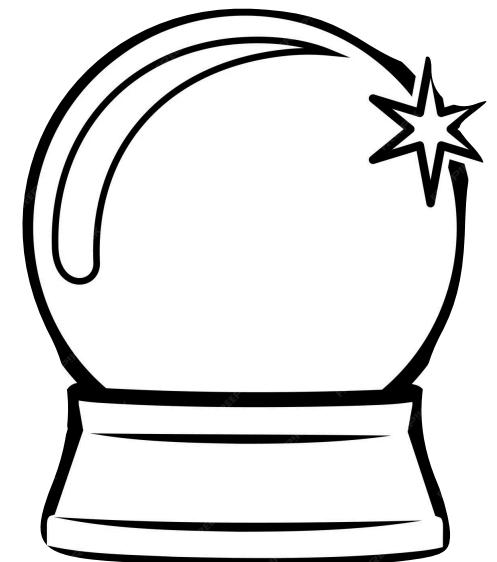
memory
(size = 120 B)

SSD
(size = few GBs)



page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

Let's think together



gives us the **exact position** of the **qualifying entries**

Understanding the implications of access granularity

query: $x < 4$



80 B (2 I/O)

memory
(size = 120 B)

SSD
(size = few GBs)

oracle

oracle

2 3

output buffer

6 7 9 11 2

8 9 6 3 11

6 7 9 11 2

8 9 6 3 11

7 10 4 1 5

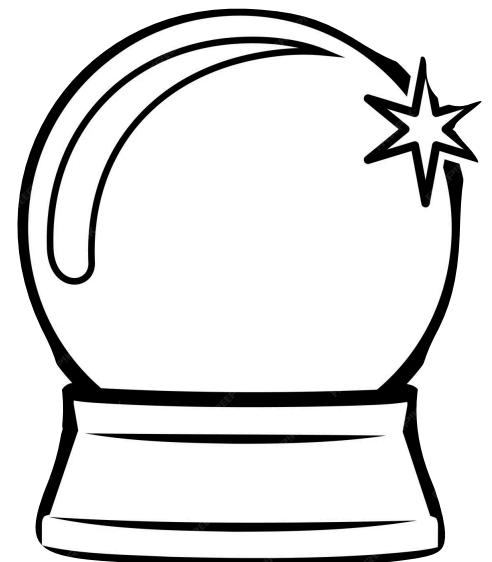
page 1

page 2

page 3

page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

Let's think together



Understanding the implications of access granularity

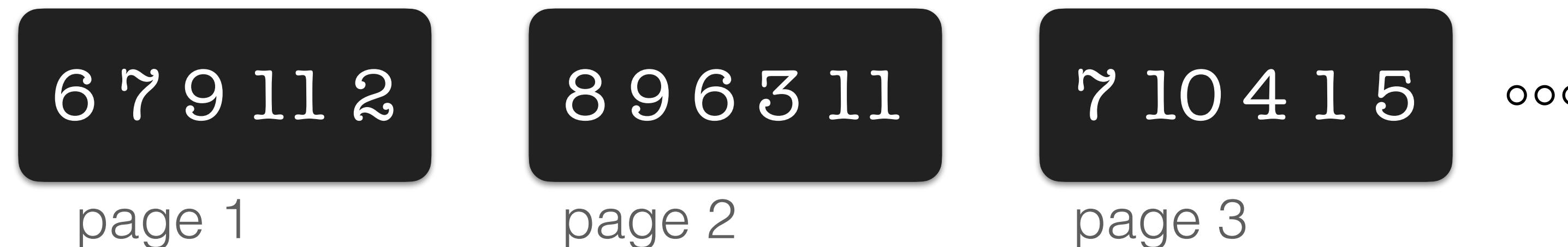
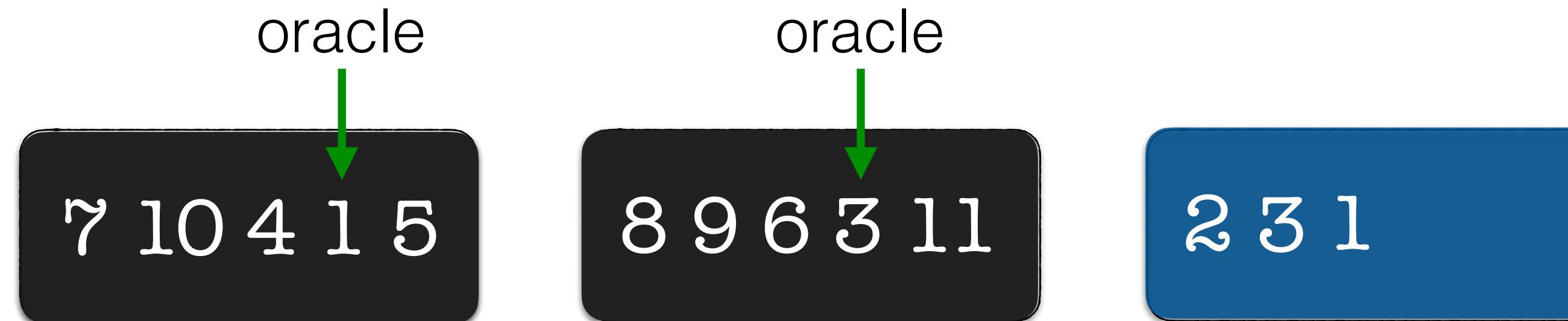


80 B (2 I/O)

gives us the **exact position** of the **qualifying entries**

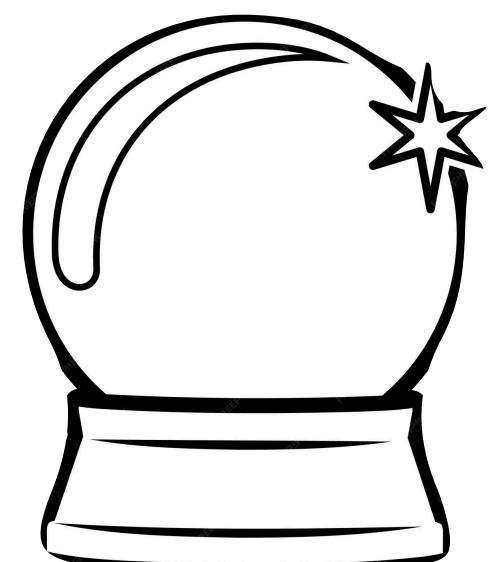
memory
(size = 120 B)

SSD
(size = few GBs)



page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

Let's think together



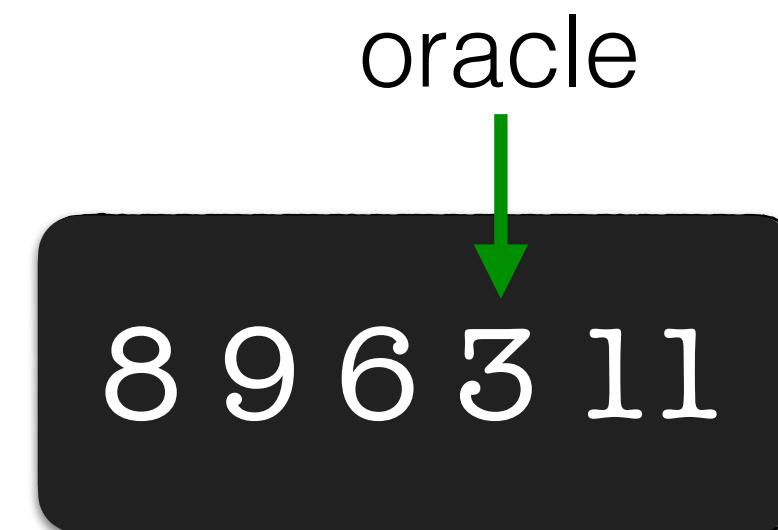
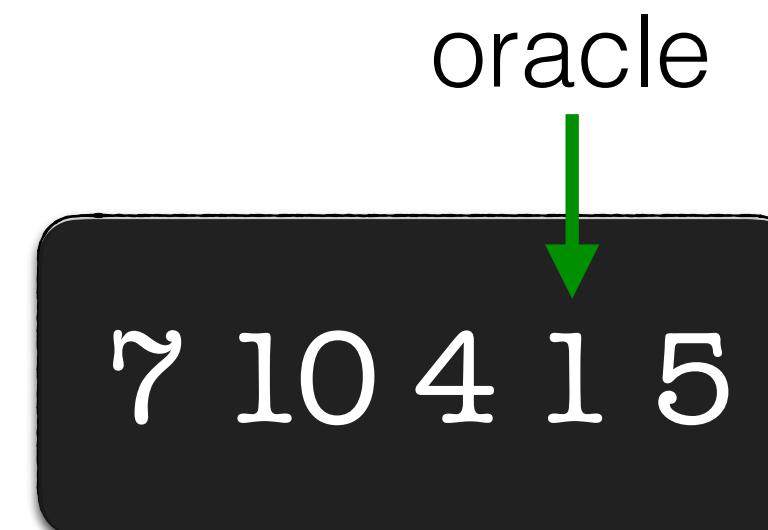
gives us the **exact position** of the **qualifying entries**

memory

(size = 120 B)

SSD

(size = few GBs)



output buffer



page 1



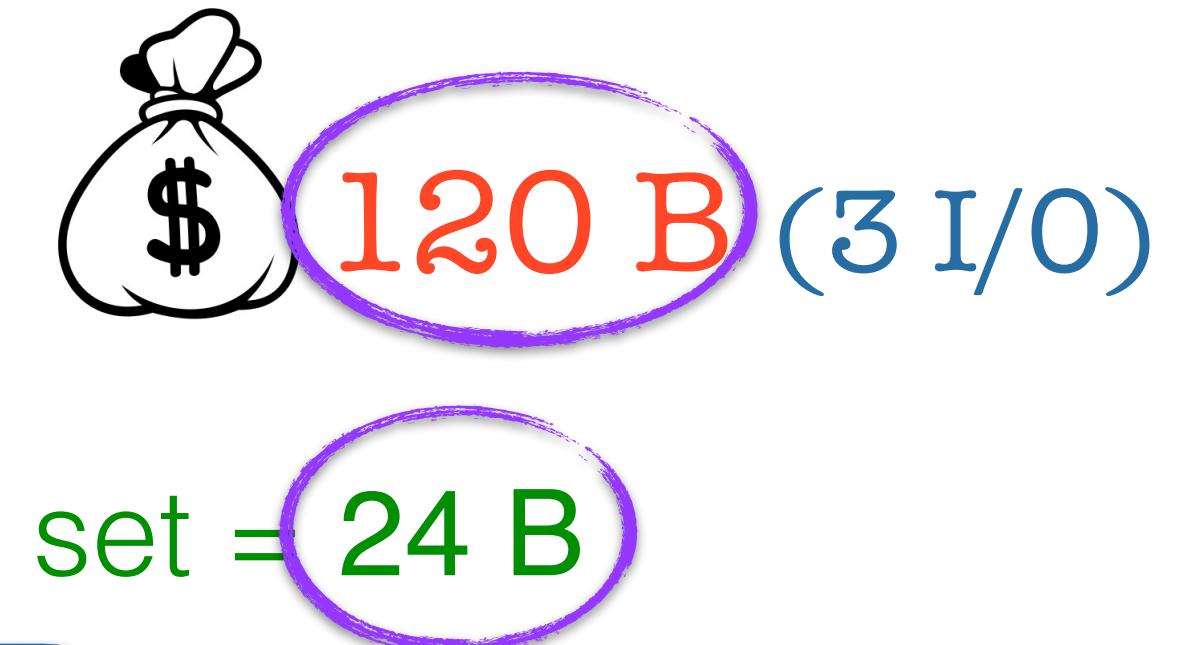
page 2



page 3

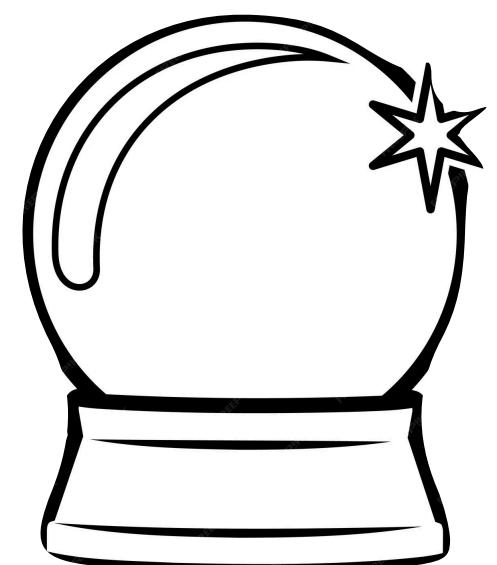
ooo

page size = 5 x sizeof(integer) = 5 x 8 B = 40 B



Was the index helpful?

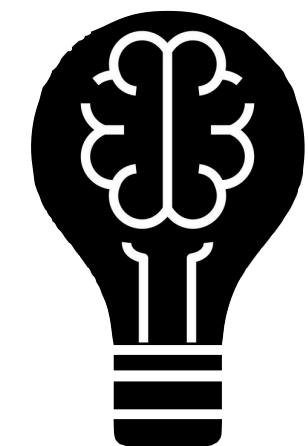
Let's think together



gives us the **exact position** of the **qualifying entries**

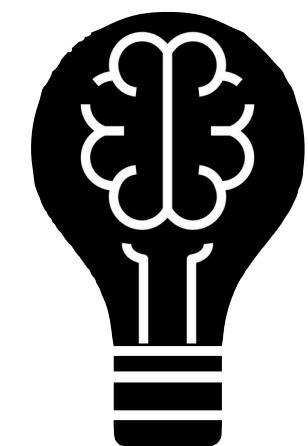
memory
(size = 120 B)

SSD
(size = few GBs)



Thought Experiment 4

For which **query** would an oracle **help us?**



Thought Experiment 5

When do we **use an oracle** and when **not?**

6 7 9 11 2

page 1

8 9 6 3 11

page 2

7 10 4 1 5

page 3

ooo

page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

Storing data

Things to keep in mind

How we store (write) data heavily determines the performance of the system

Disk is 6 orders of magnitude slower than CPU

SSDs are 4 orders of magnitude slower

Memory is 3 orders of magnitude slower

Random vs. Sequential access

So, be VERY careful!

Avoid disk accesses (reads/writes) whenever possible

I/Os to secondary storage is *always slow!*

Sequential access

read **each block exactly once**; process it; discard it; read next block

modern hardware can predict and **prefetch**; maximize performance

Random access

read a block; process it **partially**; discard it; may **read** the block **again**
often leads to **read amplification**

Random vs. Sequential access

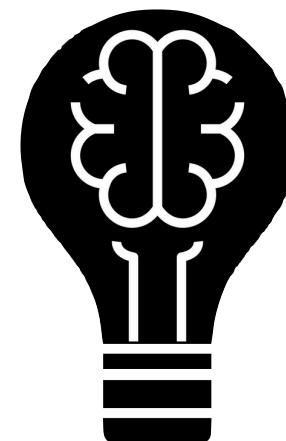
So, be VERY careful!

Sequential access

read **each block exactly once**; process it; discard it; read next block
modern hardware can predict and **prefetch**; maximize performance

Random access

read a block; process it **partially**; discard it; may **read** the block **again**
often leads to **read amplification**



Thought Experiment 6

Are **random accesses** always **bad**?



Not, if we can avoid a large number of I/Os

Project 1

Testing the waters!

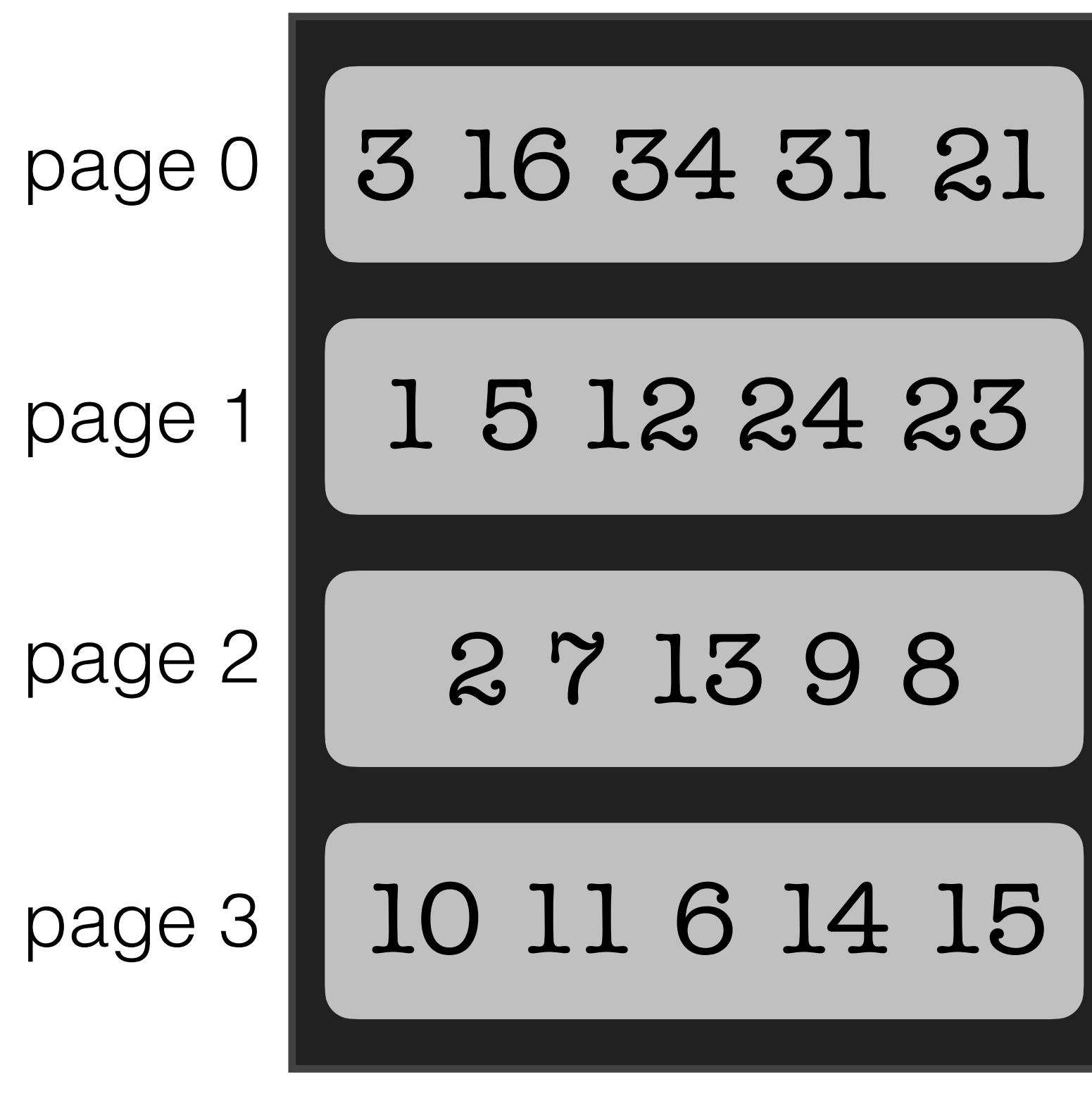
Implementing a Simple Zone Map

zone map: A **light-weight index** data structure that helps in avoiding expensive I/Os to secondary storage

Project 1

Testing the waters!

Implementing a Simple Zone Map



data on disk is stored in **files** (heap, sorted)

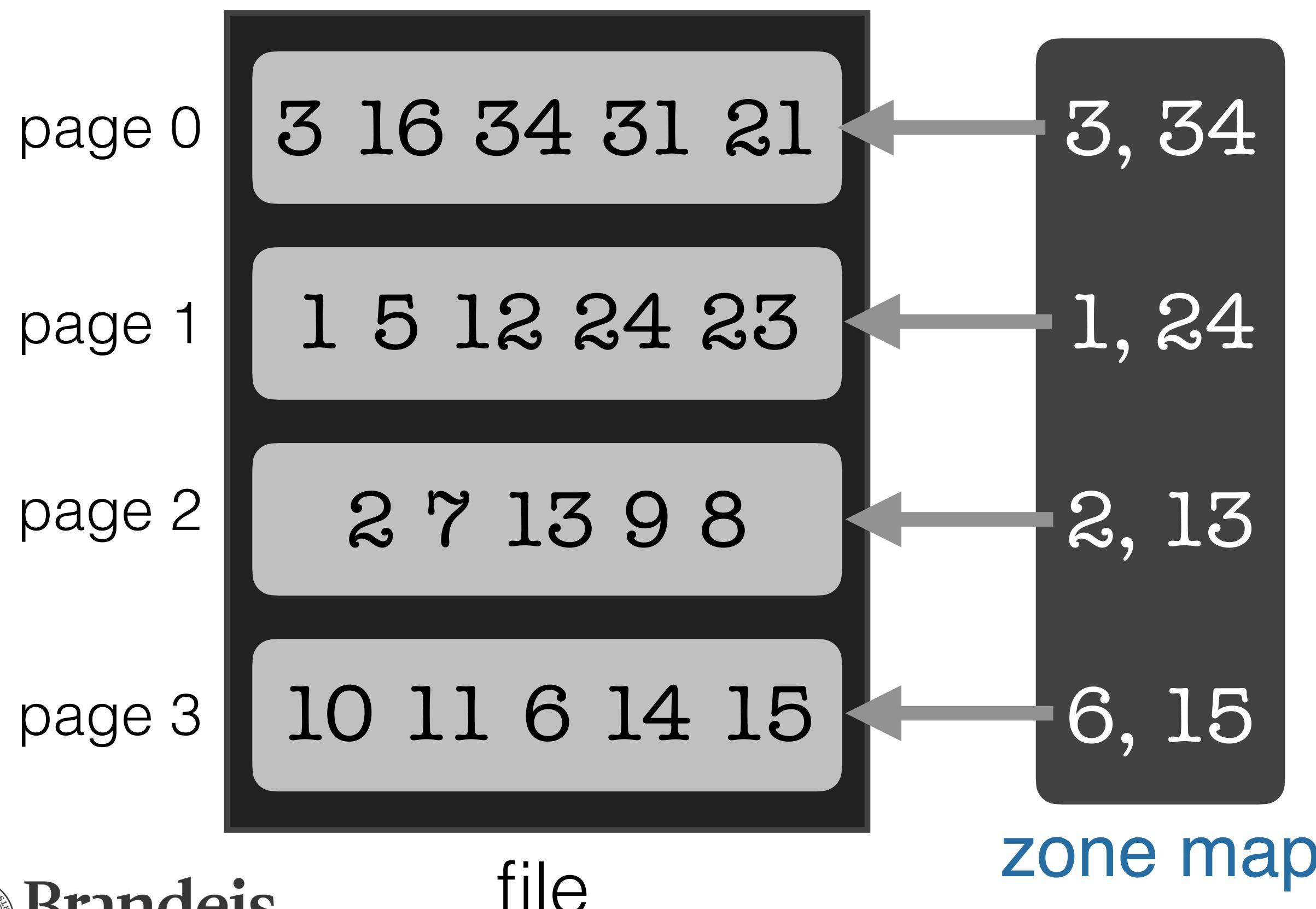
a file is a collection of **pages** (blocks)

within the page there are **data entries**

Project 1

Testing the waters!

Implementing a Simple Zone Map



w/o ZM: queries take 4 I/Os

with ZM: query: $x < 4$: 3 I/Os

$x < 12$: 4 I/O

$x = 1$: 1 I/O

$x = 20$: 4 I/O

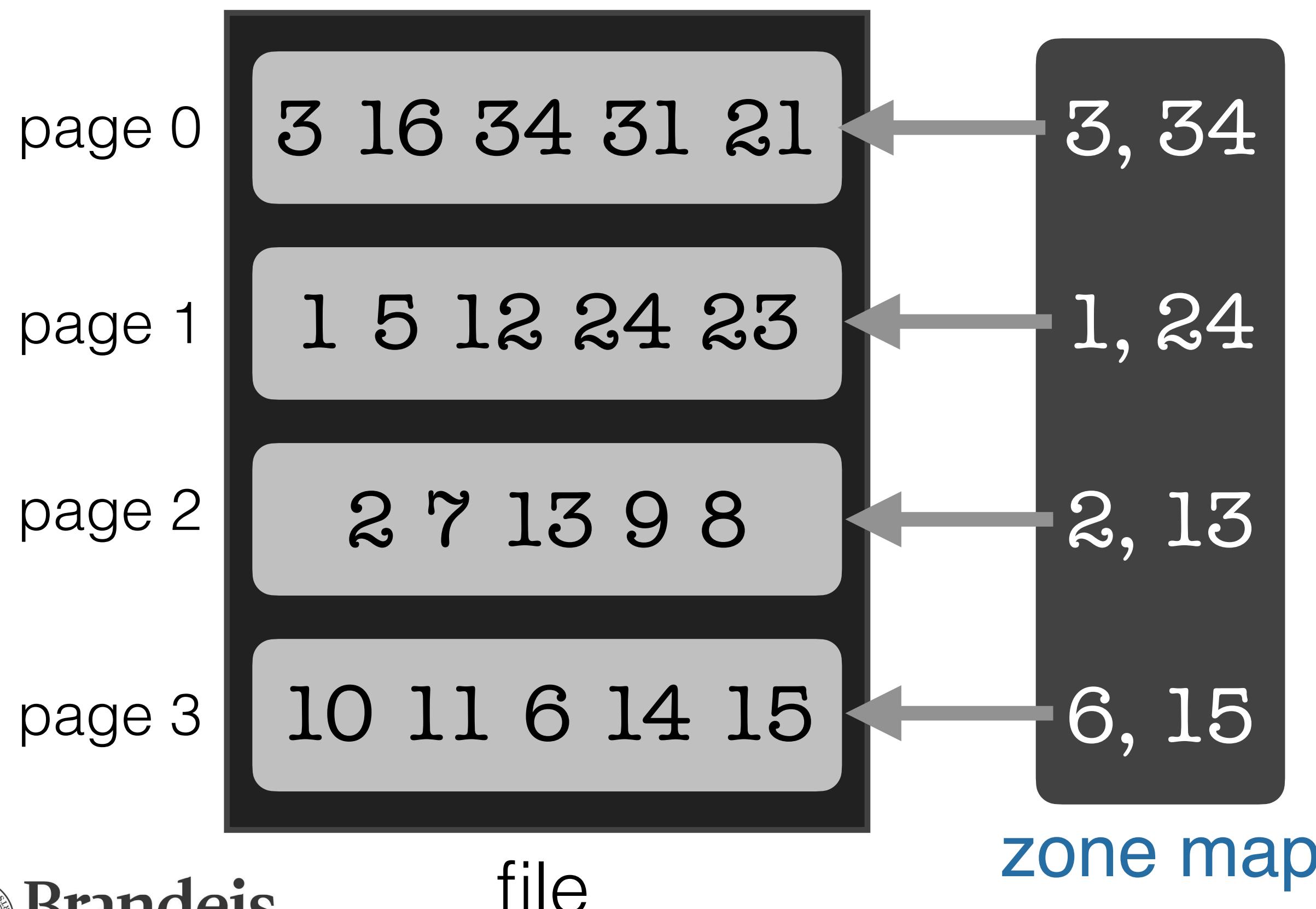
Useful for some queries;
but never harmful!



Project 1

Testing the waters!

Implementing a Simple Zone Map



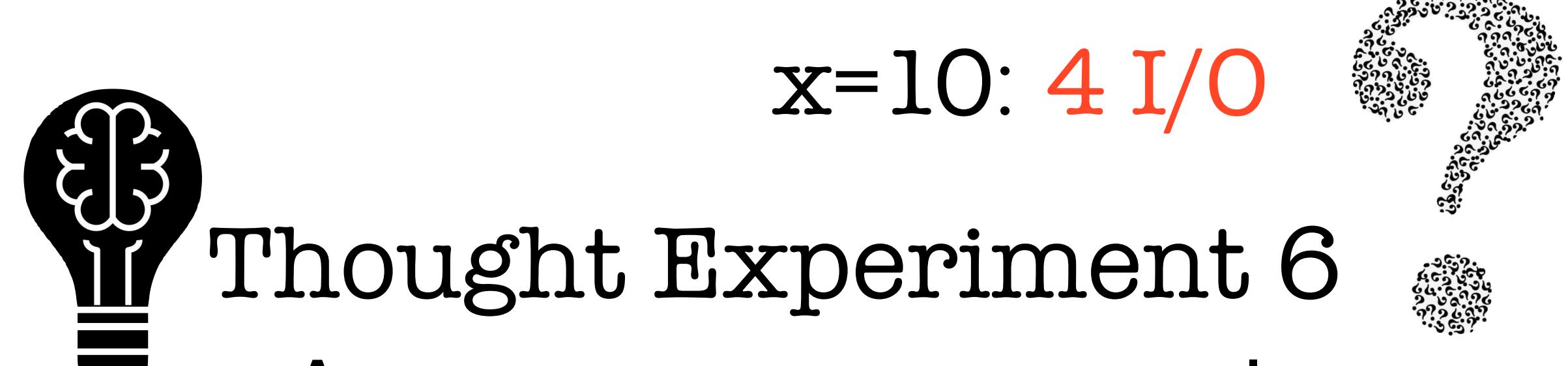
w/o ZM: queries take **4 I/Os**

with ZM: query: $x < 4$: **3 I/Os**

$x < 12$: **4 I/O**

$x = 1$: **1 I/O**

$x = 10$: **4 I/O**



Are **zone maps** more or less useful if data is **sorted**?

Readings for next class

Papers, papers, and papers

The Seattle Report on Database Research

— J. Hellerstein, M. Stonebraker and J. Hamilton
SIGMOD Record, 2022

The Seattle Report on Database Research

— D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden
SIGMOD Record, 2018



learn to read
technical papers

learn to critique
constructively

learn to prepare
slides & present

Your Lecturer

That's me!

Subhadeep Sarkar

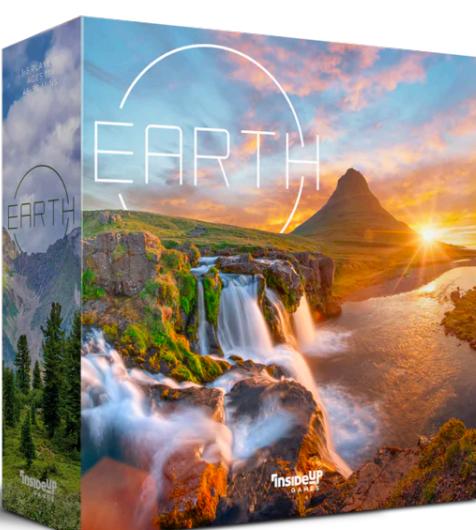
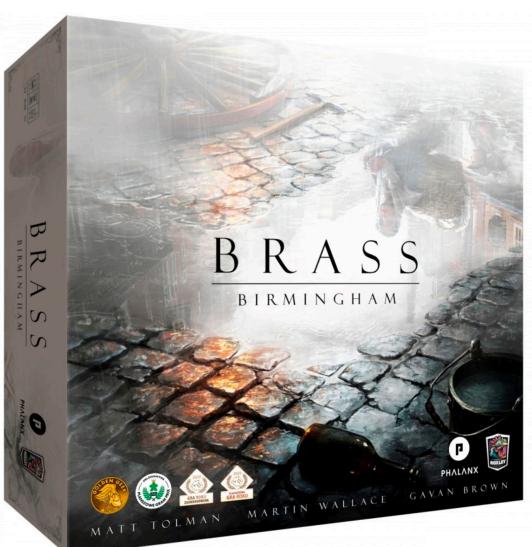
Name in Bengali: শুভদীপ সরকার

Post-doc 2: Boston University

Post-doc 1: Inria, France

PhD: Indian Institute of Technology, Kharagpur

Things I love: sports, adventure, animals, board games



Assignment submission

And grading!



Code: **PYG88X**

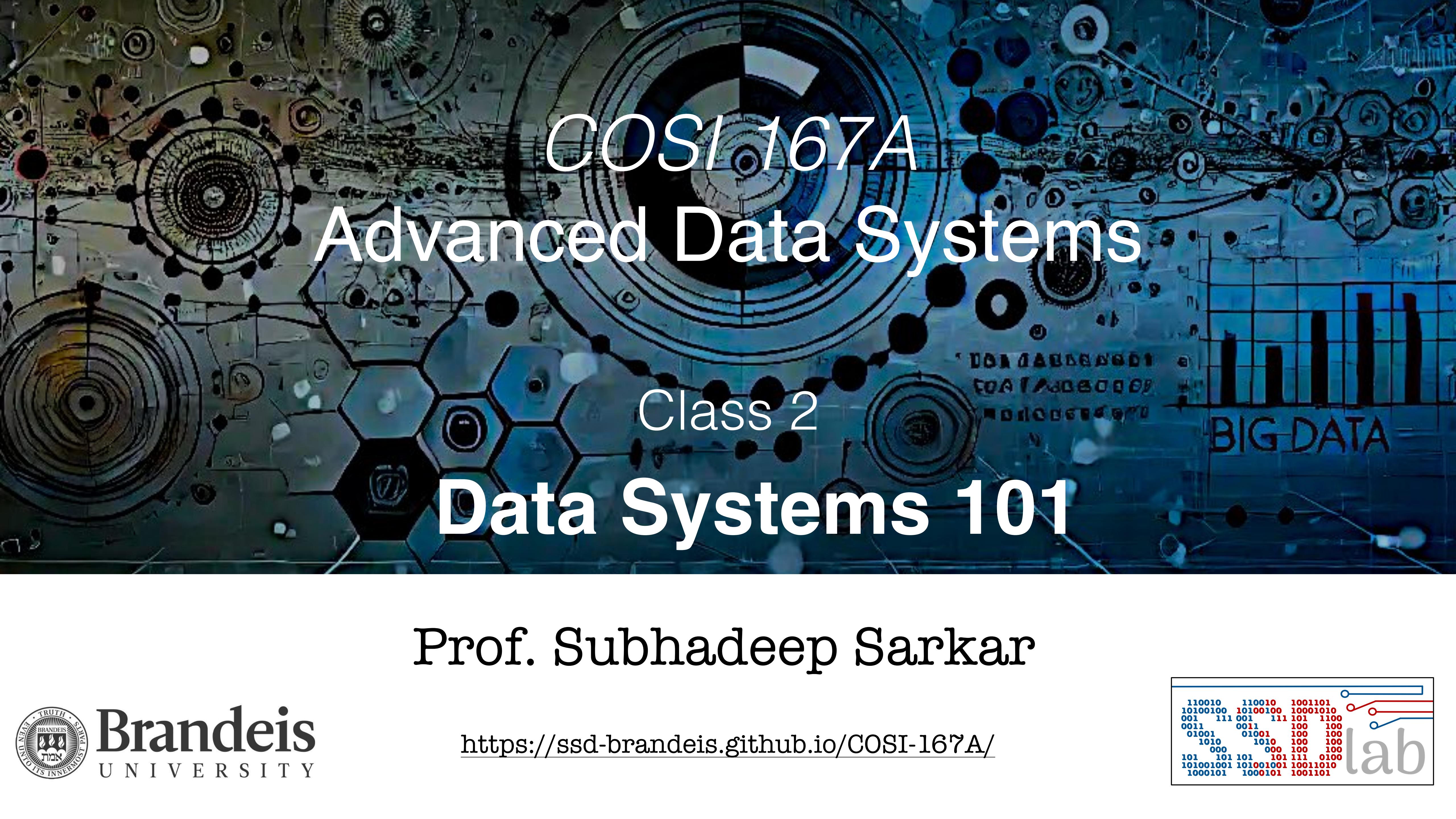
don't be late in your submissions!

Next time in COSI 167A

Intro. + Administrivia

fundamentals of data storage

introduction to row-stores and column-stores



COSI 167A Advanced Data Systems

Class 2

Data Systems 101

Prof. Subhadeep Sarkar



Brandeis
UNIVERSITY

<https://ssd-brandeis.github.io/COSI-167A/>

