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 167A

What's on the cards?

class **logistics**, **goals**, and **administrivia**

introduction to **NoSQL systems**

Project 1 details

Brandeis UNIVERSITY

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?"**

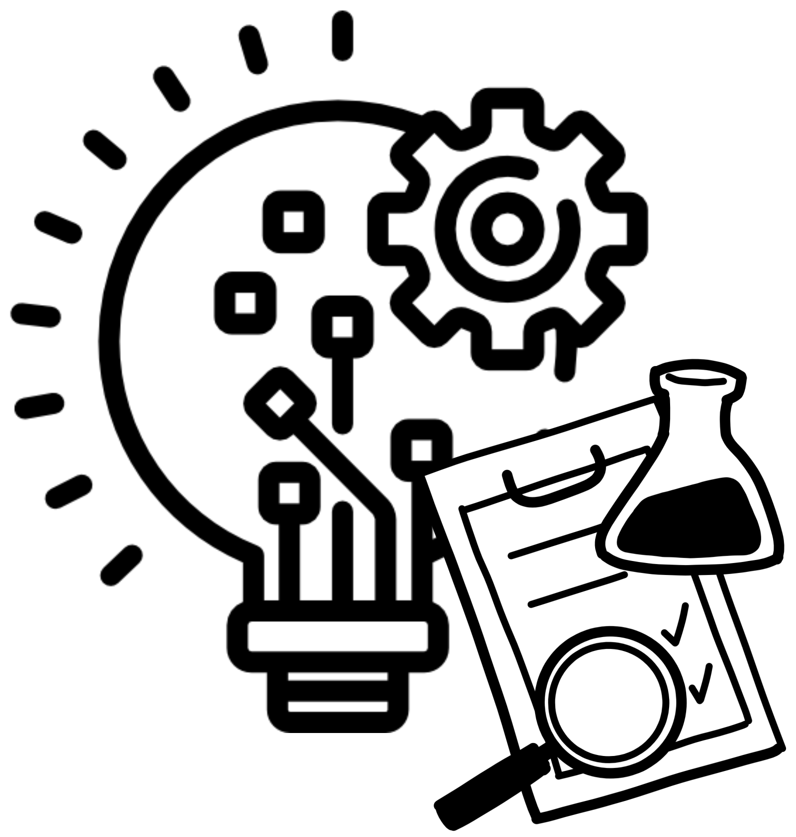
querying BIG data & querying **fast** — **Querying unstructured data, SQL?**

new **system designs** — **New application requirements = New design trends**

getting your **hands "dirty"** — **Play with large-scale, commercial storage engines**

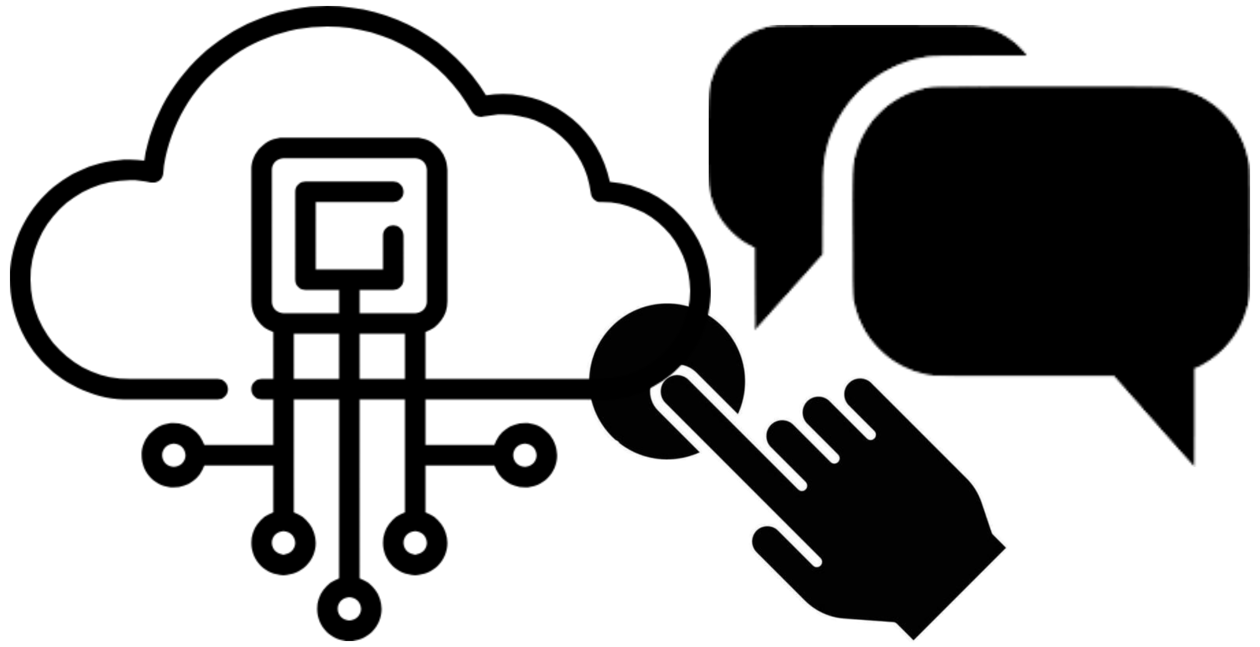
Recap: Class **Philosophy**

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!

Brandeis UNIVERSITY

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!

projects **55%**

Project 1 **15%**

individual project

released **tomorrow**; due on: **Sep 20**

more details **later in the class**

https://ssd-brandeis.github.io/COSI-167A/assets/PAs/PA1/PA-1.pdf

Class project **40%**

group (of 2) project

4 milestones

- project **proposal** **5%**
- **mid-term** report **5%**
- project **presentation** **10%**
- **code review** **20%**

Brandeis UNIVERSITY

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

data mining business datasets
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!

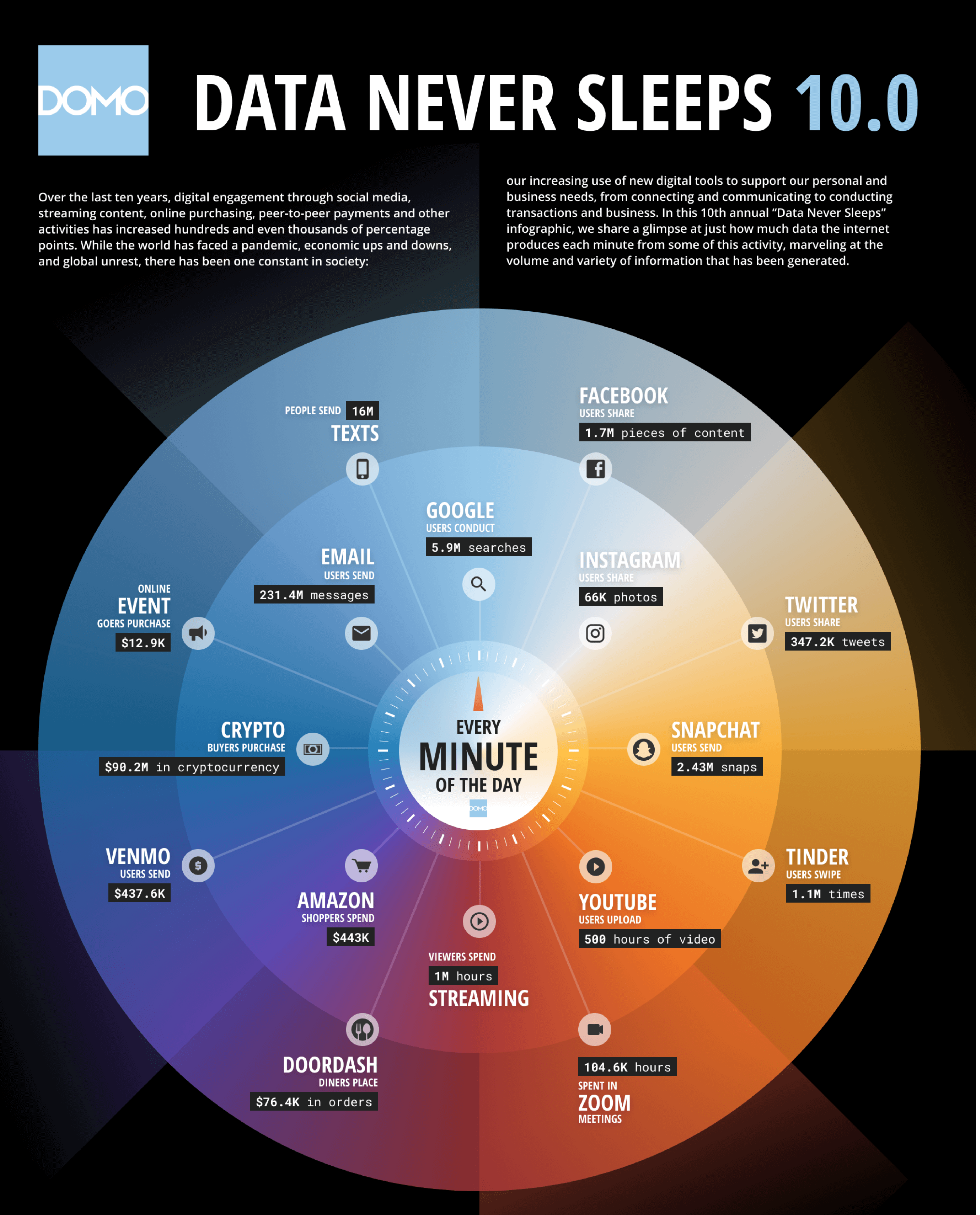
experimental **physics** (IceCube, CERN)
neuroscience
biology

data mining business datasets
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

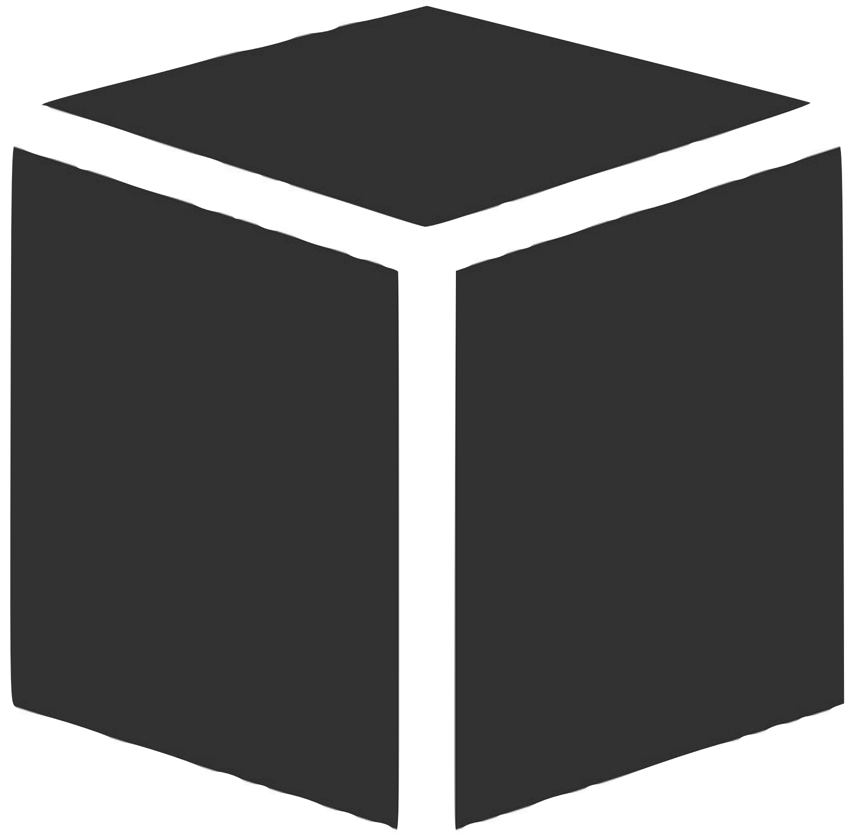


Managing **big data**

is a mammoth undertaking!

requires

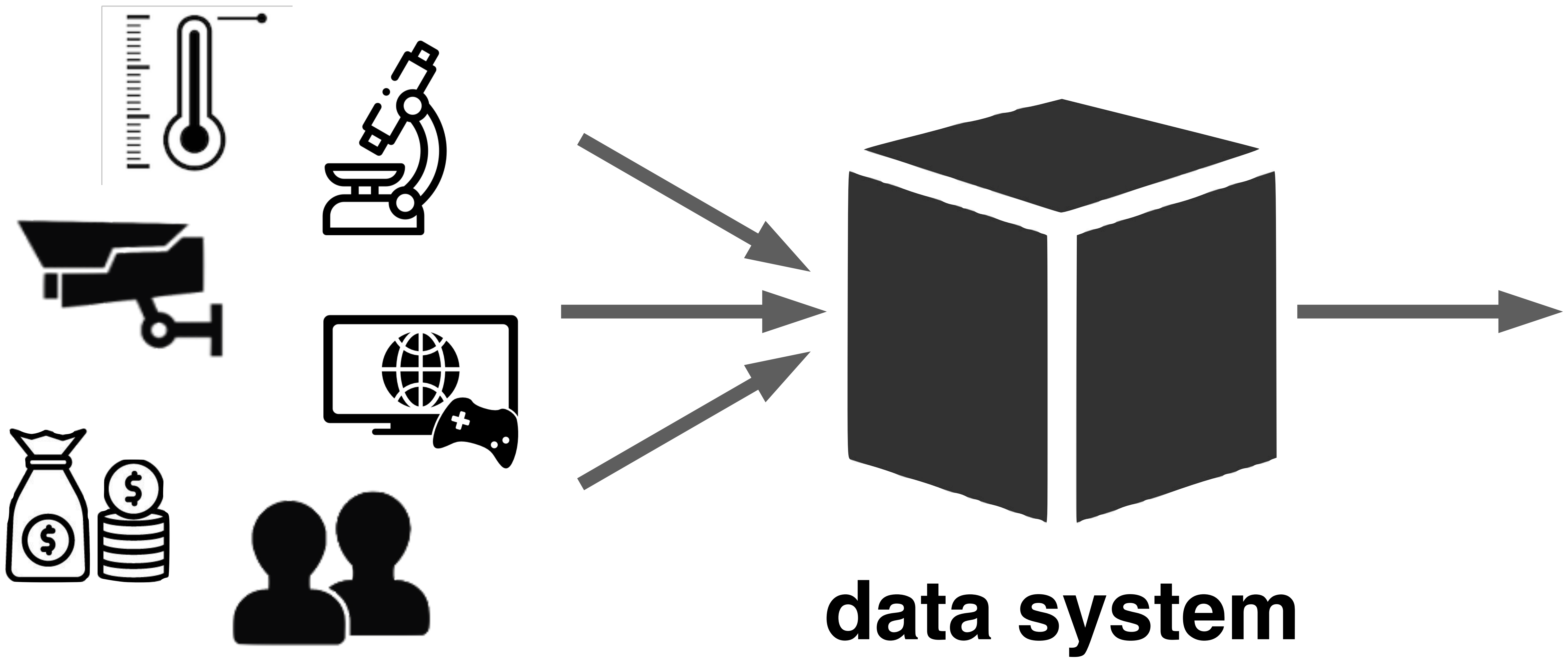
specialized systems



data system / data store / DB kernel

Data systems

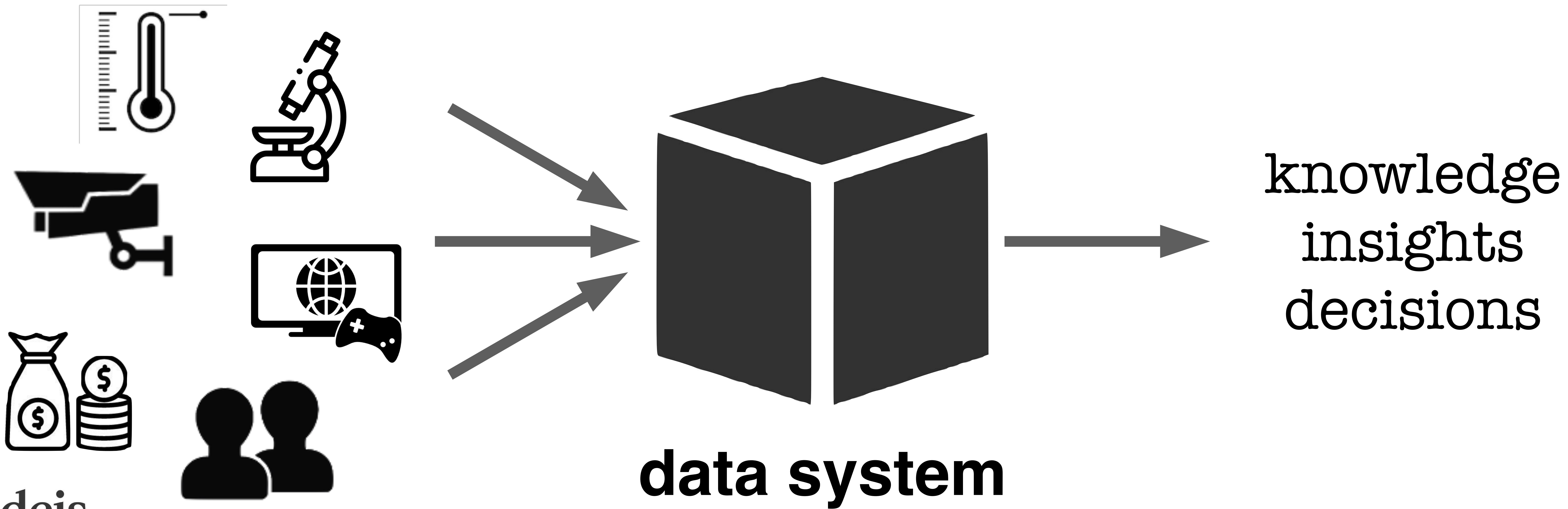
What are they, really?



Data systems

What are they, really?

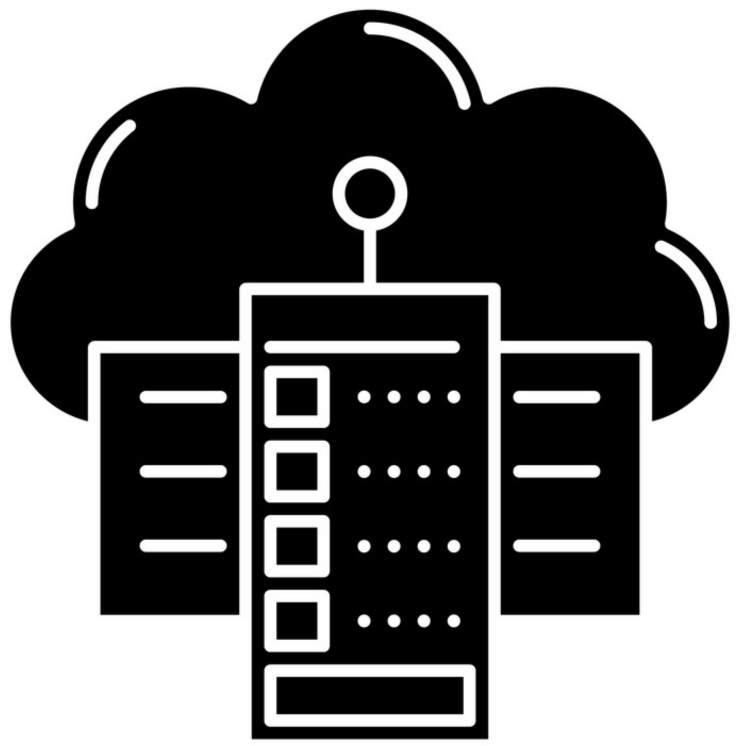
A data system is an **end-to-end software system** that is responsible for storing data and providing access to the data through **efficient data movement.**



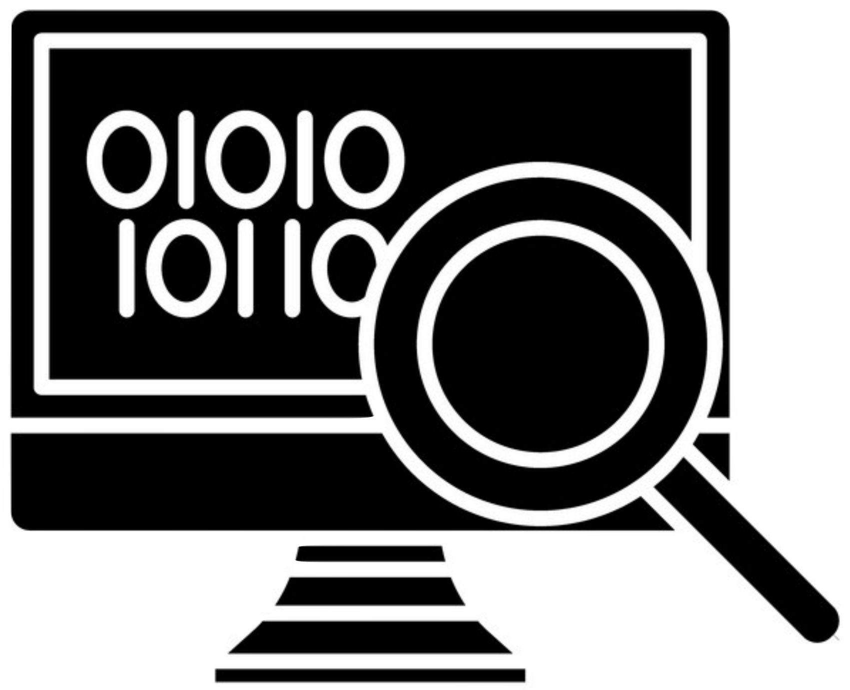
Data systems

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

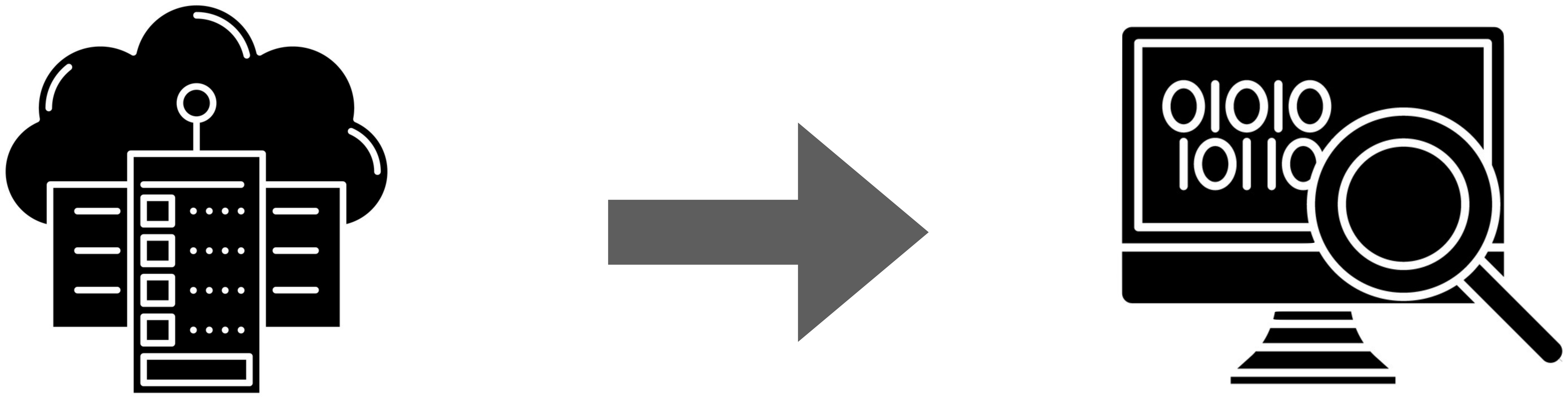


how fast we can access and analyze data

Brandeis UNIVERSITY

Data systems

What are they, really?



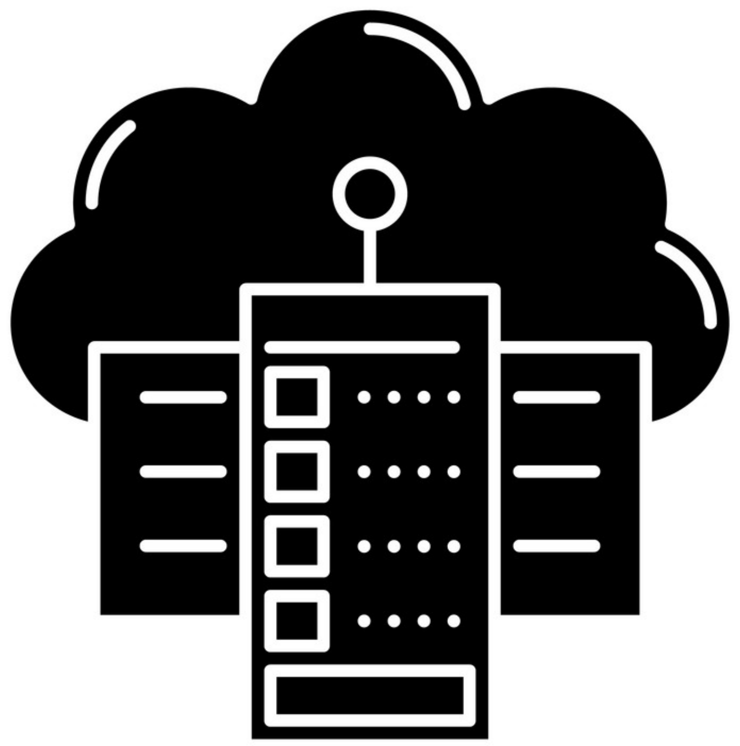
how we **store** data → how fast we can **access** and **analyze** data

70-80% of the **workload execution cost** comes from **data movement** | **20-30%** of it comes from **in-memory processing**

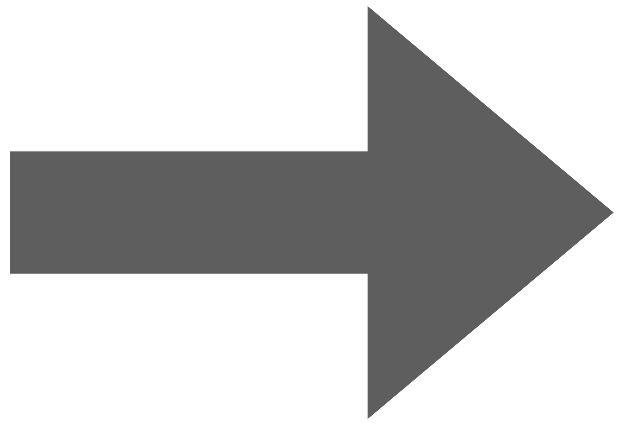
Brandeis UNIVERSITY

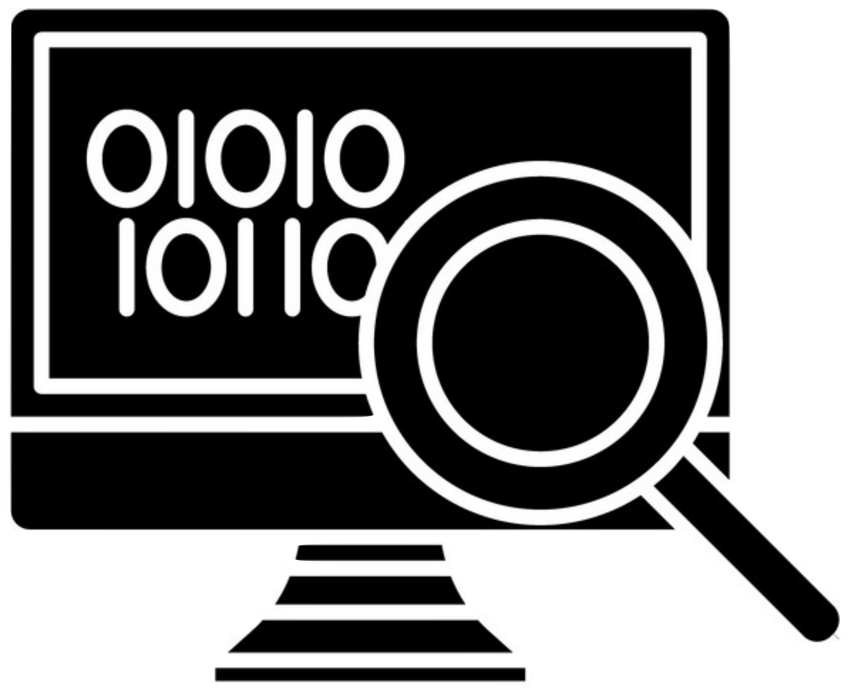
Data systems

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**

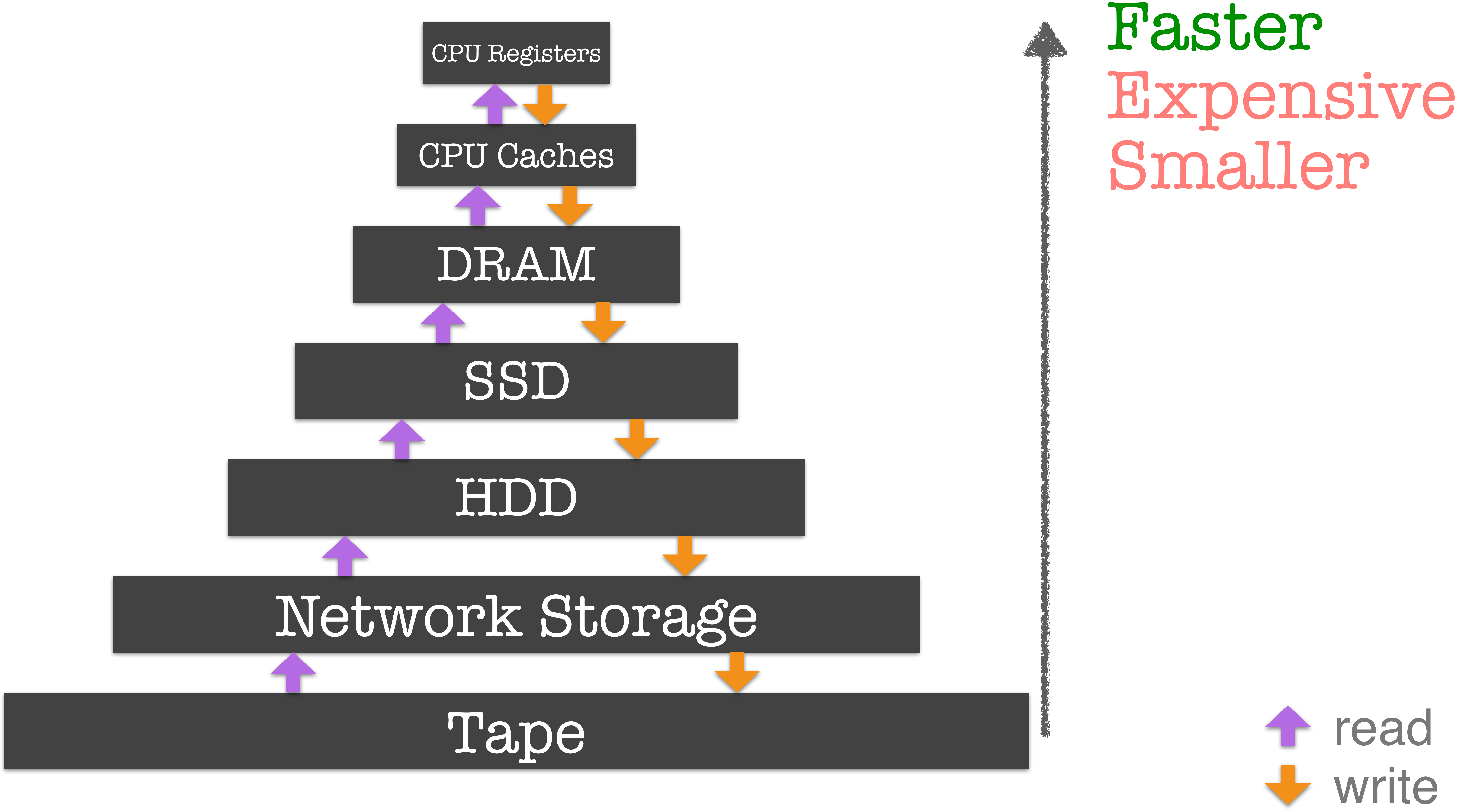
20-30% of it comes from **in-memory processing**

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

Storage hierarchy

Storage hierarchy

How data moves!



Storage hierarchy

How data moves!

Volatile

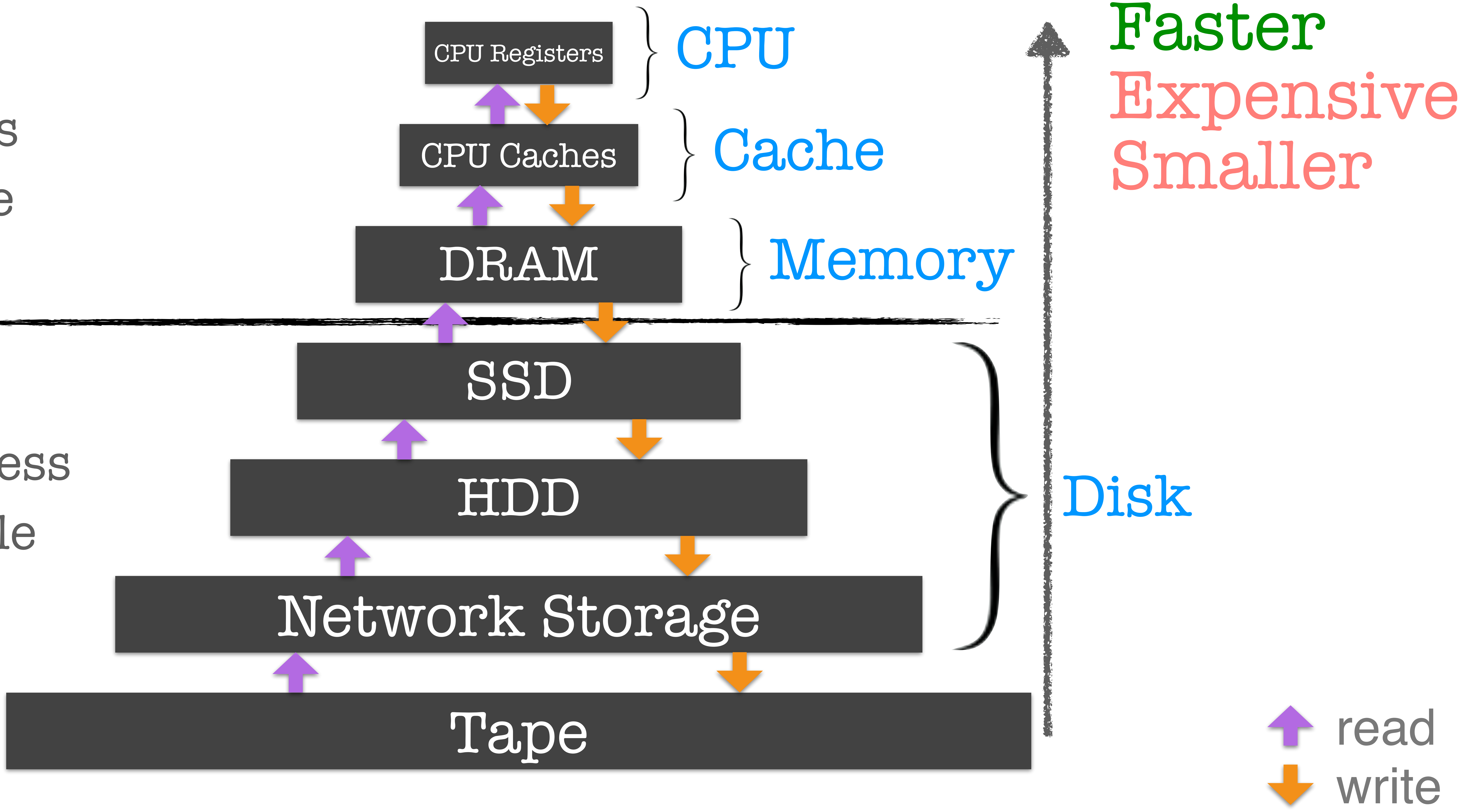
Random access

Byte accessible

Non-volatile

Sequential access

Block accessible



Brandeis UNIVERSITY

Storage hierarchy

How data moves!

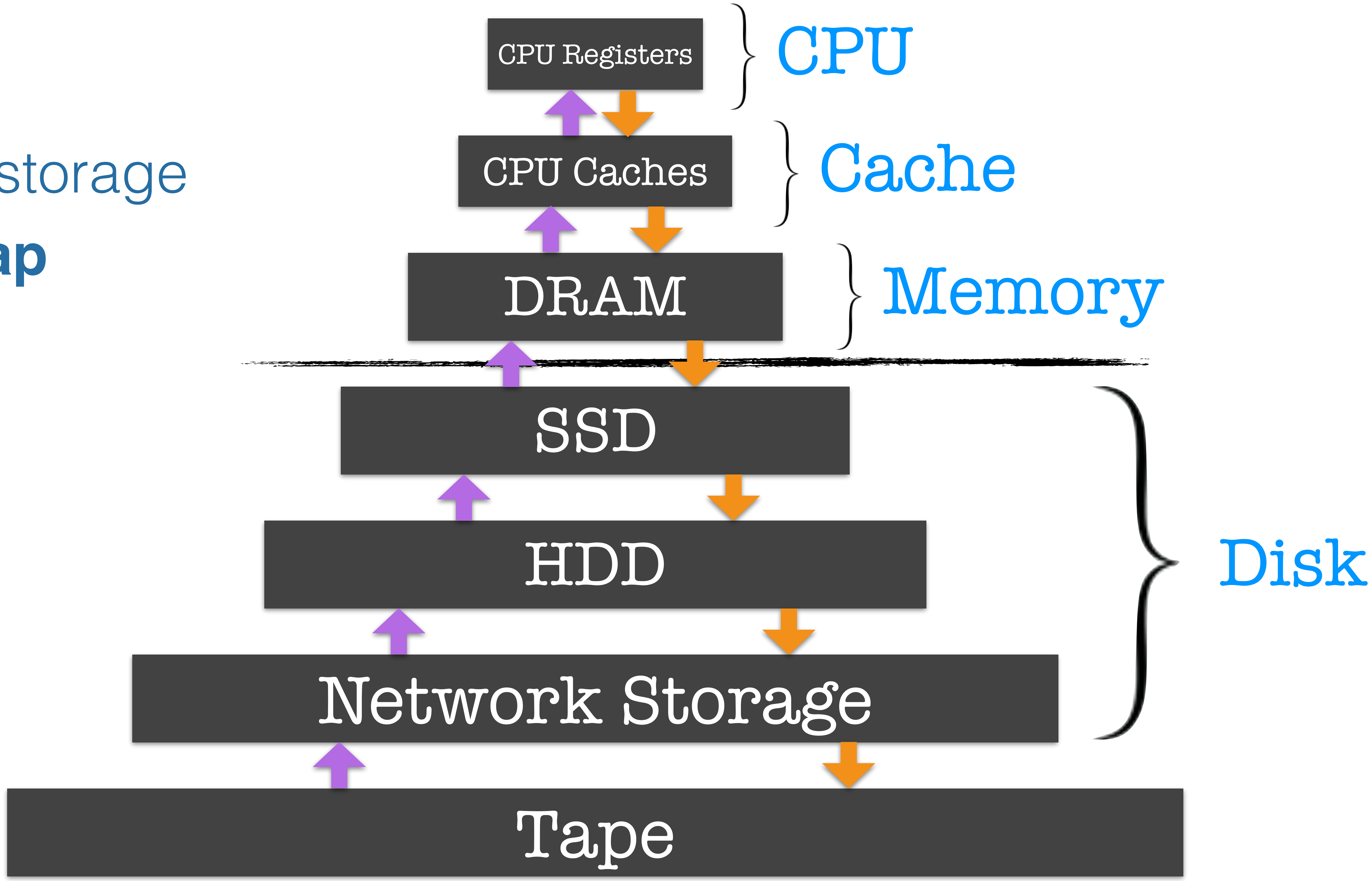
What is a **tape**?

sequential-only magnetic storage

super-slow but **super-cheap**

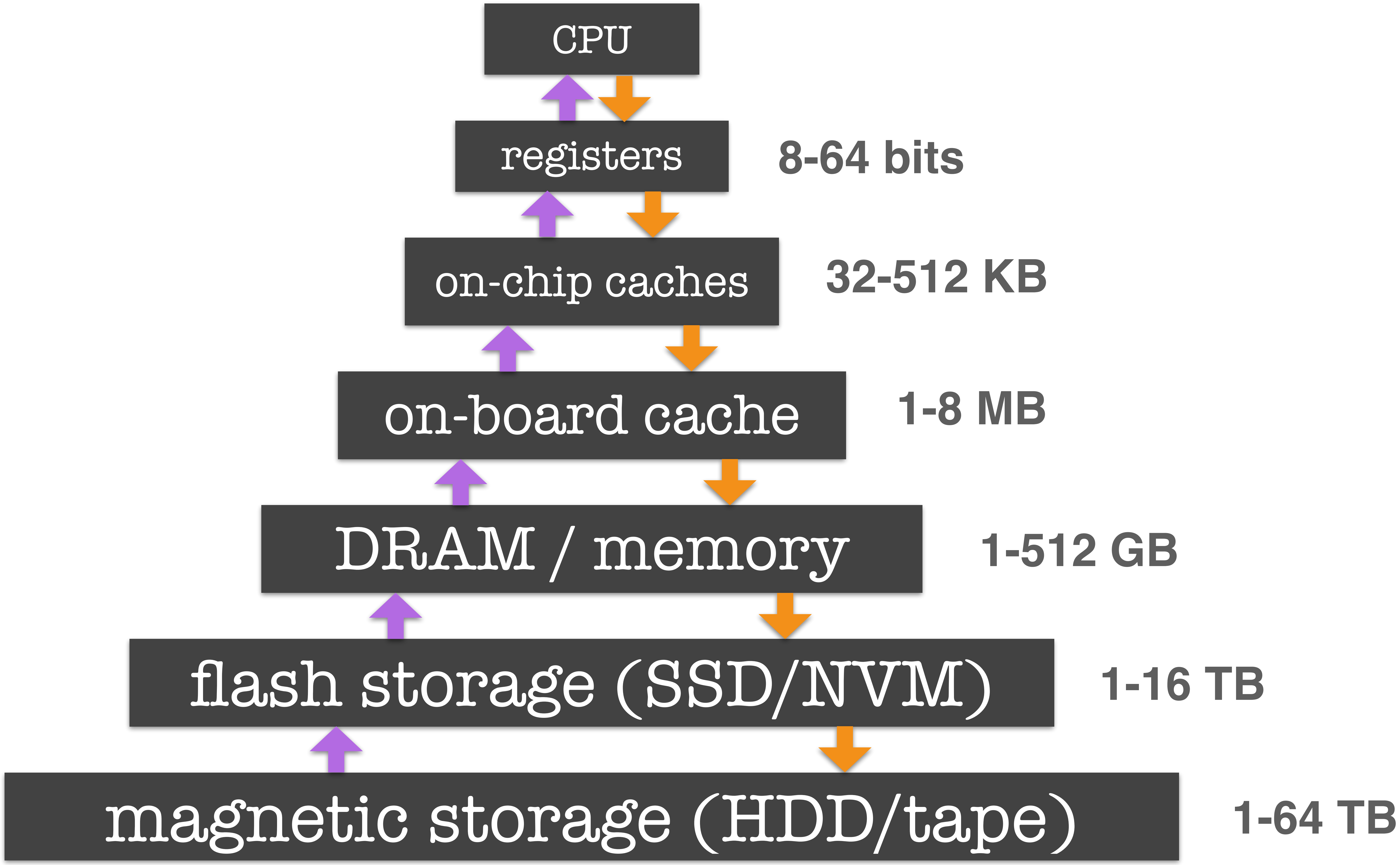
still a **multi-billion** industry

45TB @ $150



Storage hierarchy: updated

How data moves!



Brandeis UNIVERSITY

Latency numbers

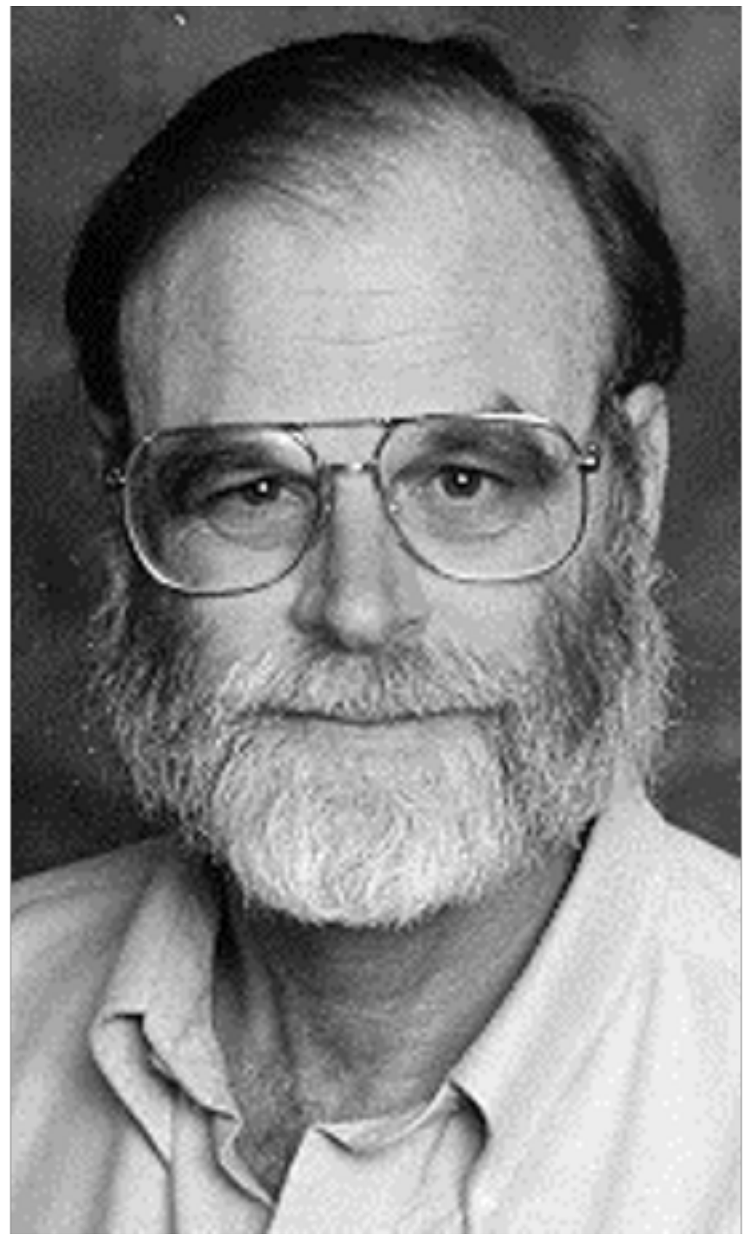
How fast is access?

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

Latency numbers

How fast is access?

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

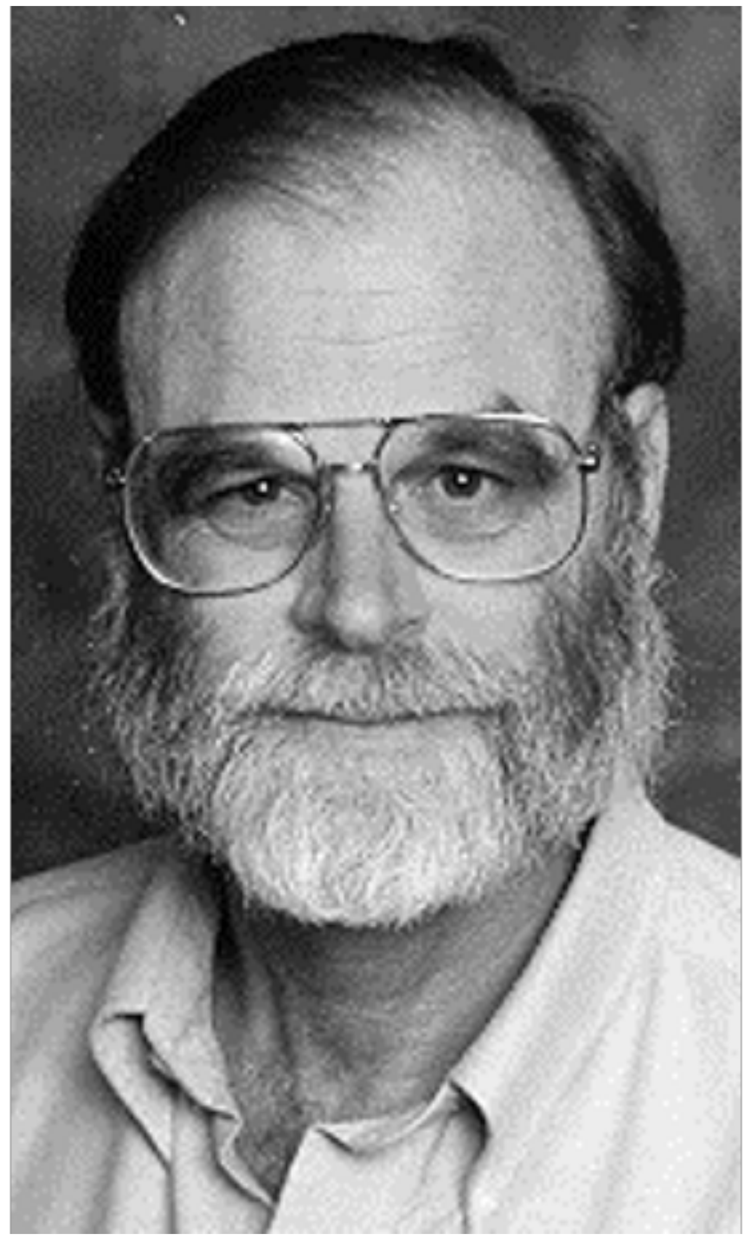


Jim Gray, IBM, Microsoft
ACM Turing Award 1998

Brandeis UNIVERSITY

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

Try not to jump the wall

computations happen here

CPU

register

on-chip cache

on-board cache

memory

flash storage

magnetic storage

Thought Experiment 1

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

cache miss!

Memory wall

Try not to jump the wall

computations happen here

CPU

register

on-chip cache

10 ns — on-board cache

100 ns — memory

flash storage

magnetic storage

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

Try not to jump the wall

computations happen here

- CPU
- register
- on-chip cache
- 10 ns — on-board cache
- 100 ns — memory
- flash storage
- magnetic storage

be careful when you go below the green line

Thought Experiment 2

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

memory miss!

Memory wall

Try not to jump the wall

computations happen here

Latency	Level
	CPU
	register
	on-chip cache
10 ns	on-board cache
100 ns	memory
16,000 ns	flash storage
2 ms	magnetic storage

be careful when you go below the green line

Thought Experiment 2

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

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

be VERY careful when you go below the red line

Access granularity

How much data to move

CPU

register

on-chip cache

on-board cache

memory

flash storage

magnetic storage

data movement

how we access data

how we store data

throughput of the system

extra/unnecessary data movement

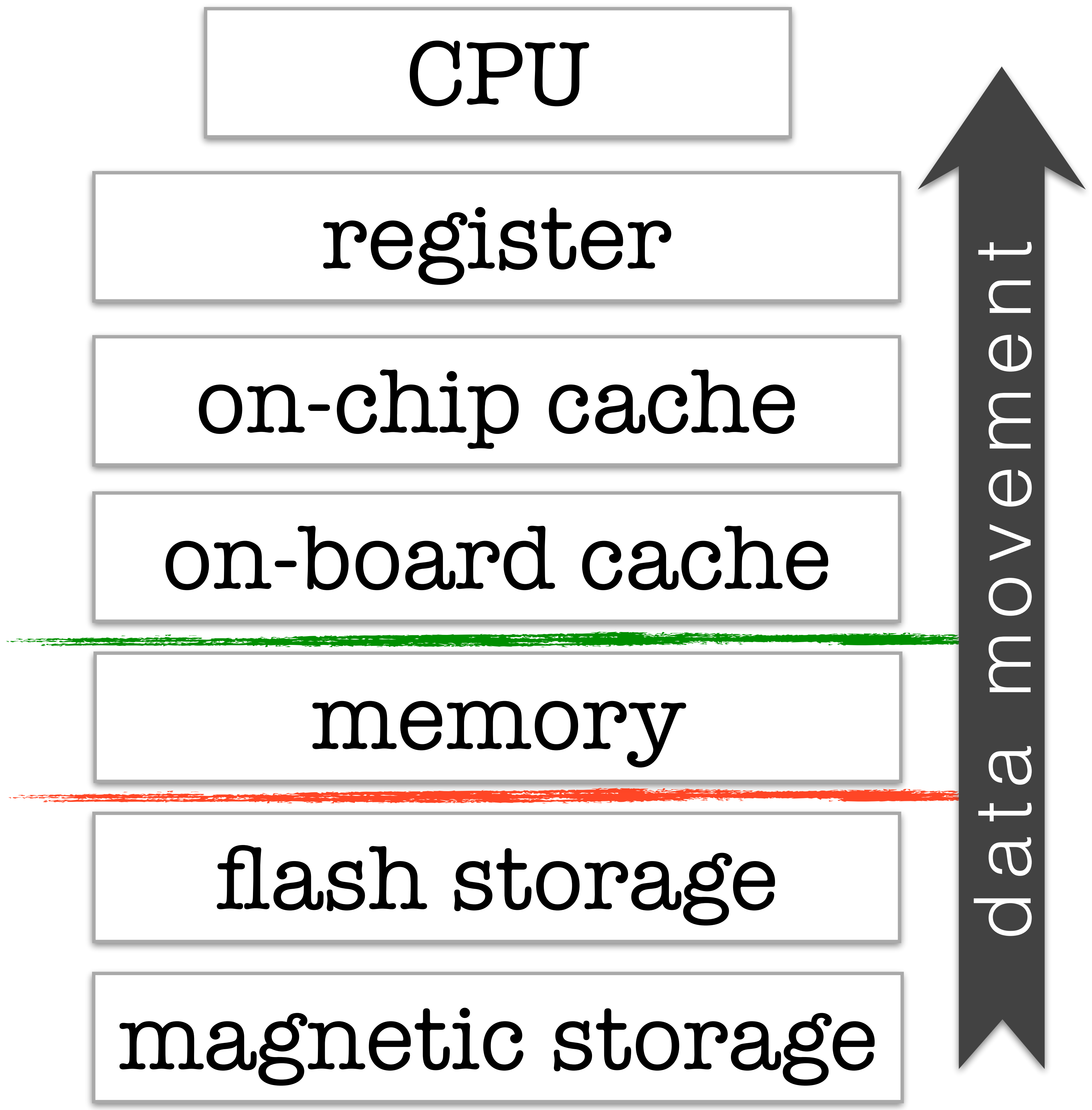
Thought Experiment 3

What causes this **unnecessary data movement**?

access granularity

Access granularity

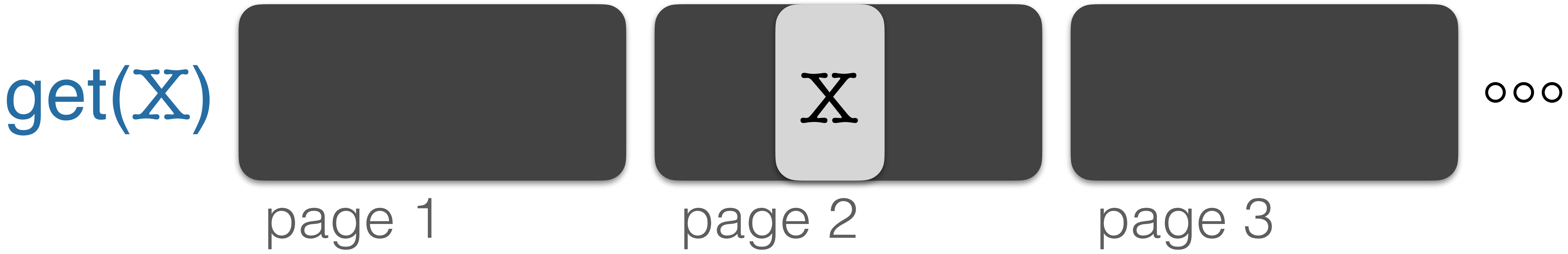
How much data to move



Thought Experiment 3

What causes this **unnecessary data movement**?

access granularity



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

memory
(size = 120 B)

40 B	40 B	40 B

SSD
(size = few GBs)

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

Brandeis
UNIVERSITY

Let's **think** together

Understanding the implications of access granularity

query: **x<4**

memory
(size = 120 B)

SSD
(size = few GBs)

page 1	page 2	page 3	
6 7 9 11 2	8 9 6 3 11	7 10 4 1 5	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**

~~binary search~~

memory
(size = 120 B)

6 7 9 11 2		

SSD
(size = few GBs)

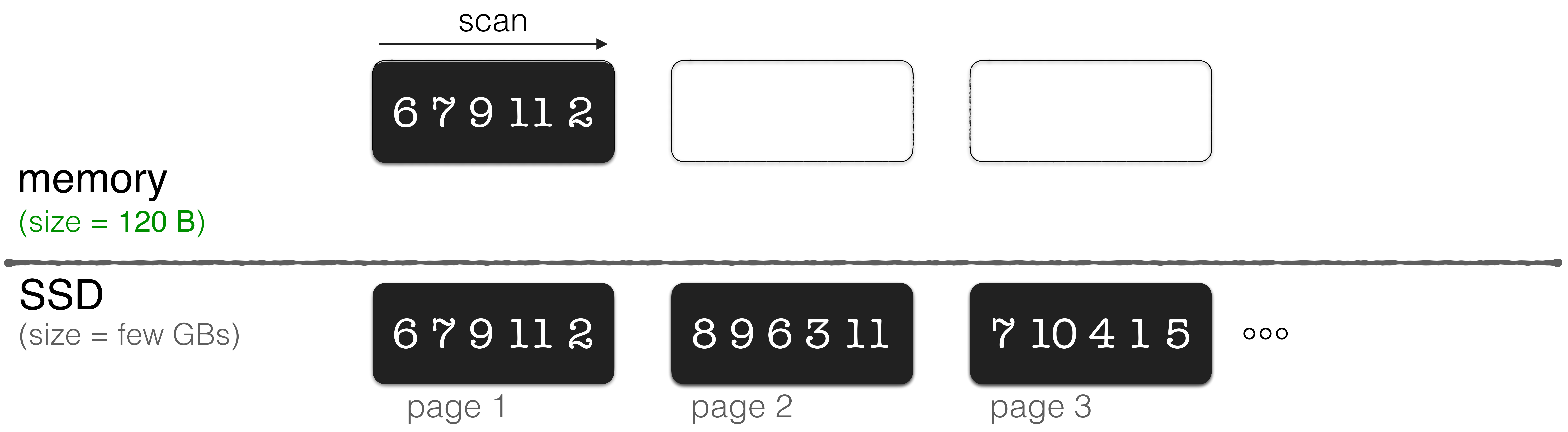
6 7 9 11 2	8 9 6 3 11	7 10 4 1 5	ooo
page 1	page 2	page 3	

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

Let's **think** together

Understanding the implications of access granularity

query: **x<4**



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

scan →

memory
(size = 120 B)

6 7 9 11 2		2
		output buffer

SSD
(size = few GBs)

page 1	page 2	page 3	
6 7 9 11 2	8 9 6 3 11	7 10 4 1 5	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

40 B (1 I/O)

scan →

memory
(size = 120 B)

6 7 9 11 2		2
		output buffer

SSD
(size = few GBs)

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

Understanding the implications of access granularity

query: x<4

40 B (1 I/O)

scan → scan →

6 7 9 11 2 | 8 9 6 3 11 | 2 3

output buffer

memory
(size = 120 B)

SSD
(size = few GBs)

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

Brandeis UNIVERSITY

Let's **think** together

Understanding the implications of access granularity

query: x<4

80 B (2 I/O)

memory
(size = 120 B)

scan → 6 7 9 11 2

scan → 8 9 6 3 11

2 3
output buffer

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

80 B (2 I/O)

scan → scan →

7 10 4 1 5 | 8 9 6 3 11 | 2 3 1

output buffer

memory
(size = 120 B)

SSD
(size = few GBs)

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

Understanding the implications of access granularity

query: x<4

$ 120 B (3 I/O)

scan → scan →

size of result set = 24 B

7 10 4 1 5	8 9 6 3 11	2 3 1

output buffer

memory
(size = 120 B)

SSD
(size = few GBs)

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

What if we had an **oracle**?

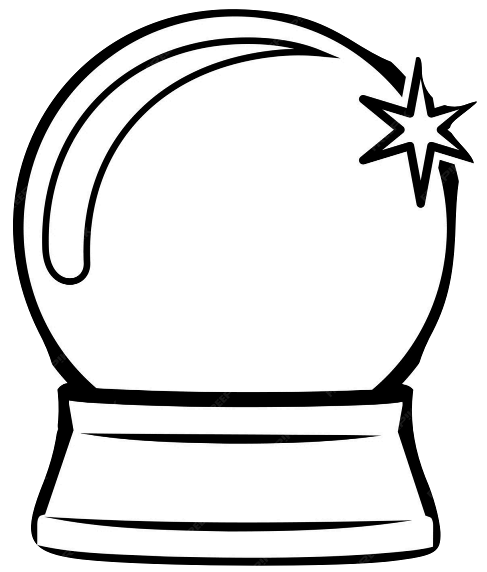
What if we had the **perfect index**?

Brandeis UNIVERSITY

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)

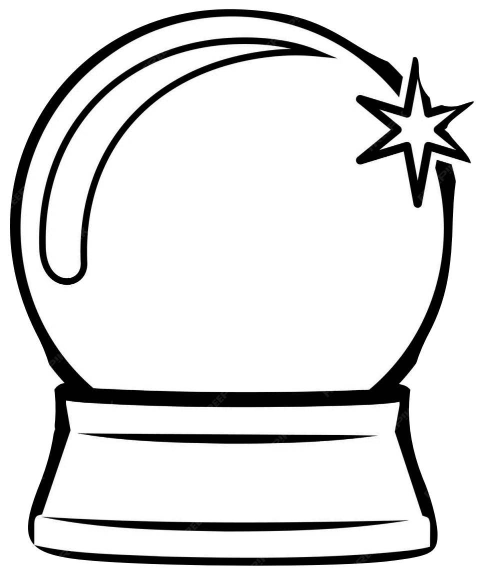
page 1	page 2	page 3	
6 7 9 11 2	8 9 6 3 11	7 10 4 1 5	ooo

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)

6 7 9 11 2		

SSD
(size = few GBs)

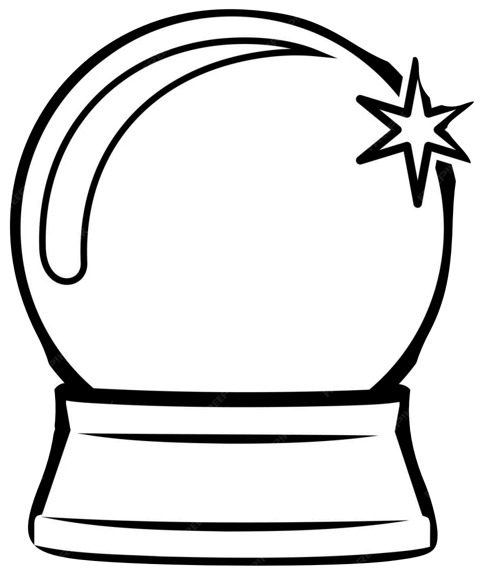
6 7 9 11 2	8 9 6 3 11	7 10 4 1 5	ooo
page 1	page 2	page 3	

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

Brandeis UNIVERSITY

Let's **think** together

Understanding the implications of access granularity



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

query: **x<4**

oracle

memory (size = 120 B)

6 7 9 11 2		

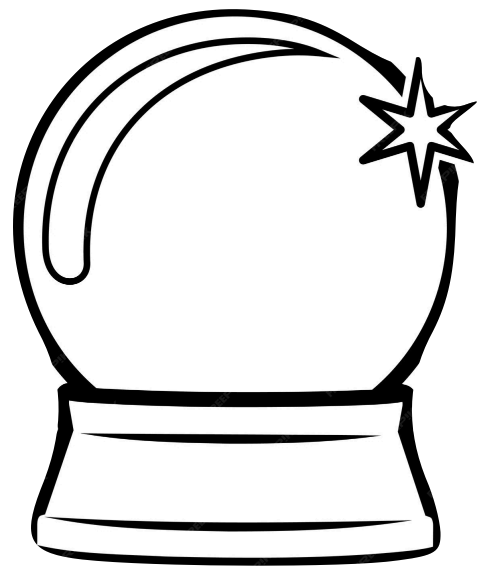
SSD (size = few GBs)

page 1	page 2	page 3	
6 7 9 11 2	8 9 6 3 11	7 10 4 1 5	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**

oracle

6 7 9 11 2

2

output buffer

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

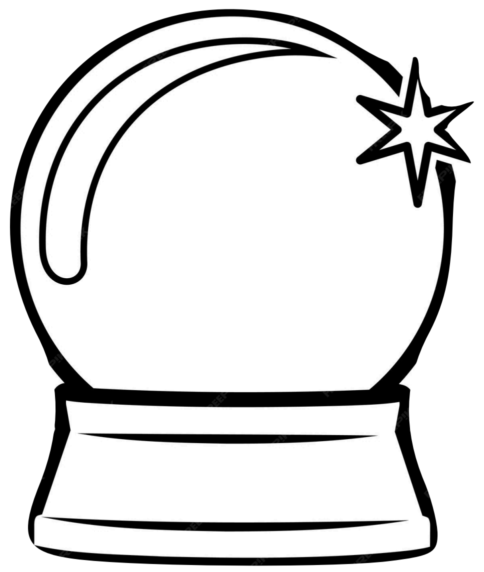
page size = 5 x sizeof(integer) = 5 x 8 B = 40 B

Brandeis UNIVERSITY

Let's **think** together

Understanding the implications of access granularity

query: **x<4**



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

40 B (1 I/O)

oracle

memory (size = 120 B)

6 7 9 11 2		2

output buffer

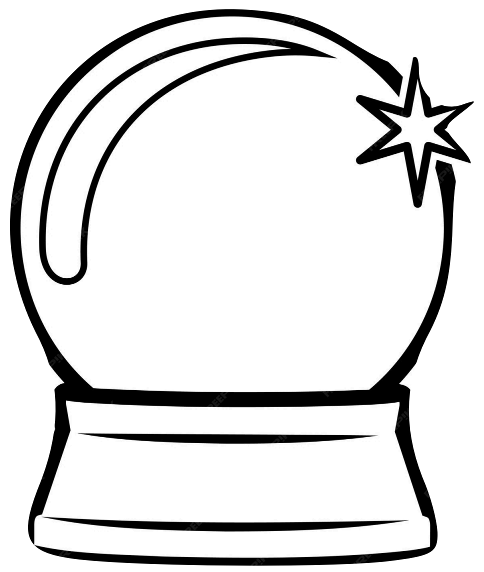
SSD (size = few GBs)

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

Understanding the implications of access granularity



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

query: x<4

$ 40 B (1 I/O)

oracle oracle

memory
(size = 120 B)

6 7 9 11 2	8 9 6 3 11	2 3
		output buffer

SSD
(size = few GBs)

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

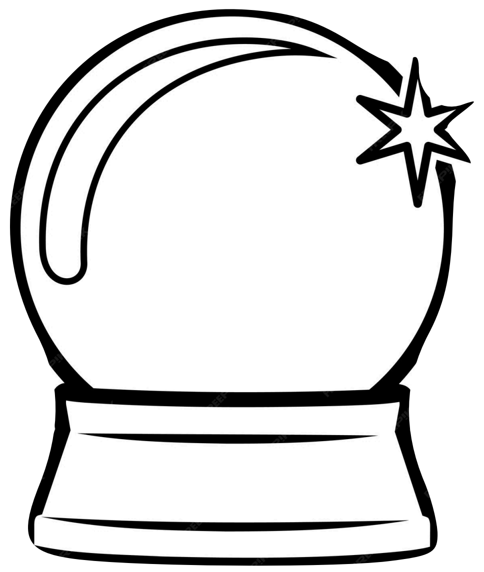
Brandeis UNIVERSITY

Let's **think** together

Understanding the implications of access granularity

query: x<4

80 B (2 I/O)



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

oracle

oracle

memory (size = 120 B)

6 7 9 11 2	8 9 6 3 11	2 3
		output buffer

SSD (size = few GBs)

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

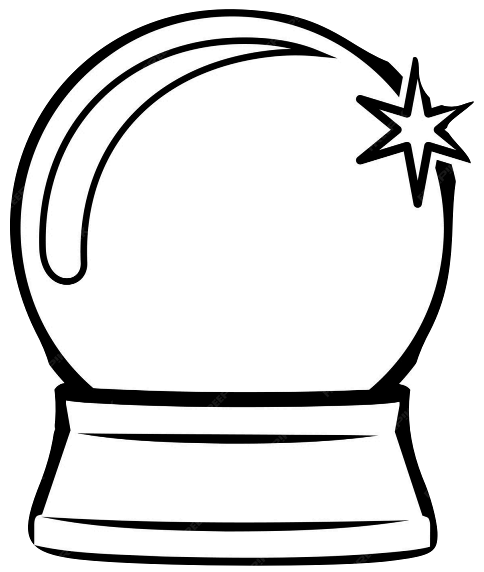
Brandeis UNIVERSITY

Let's **think** together

Understanding the implications of access granularity

query: x<4

$ 80 B (2 I/O)



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

memory (size = 120 B)

oracle ↓	oracle ↓	
7 10 4 1 5	8 9 6 3 11	2 3 1
		output buffer

SSD (size = few GBs)

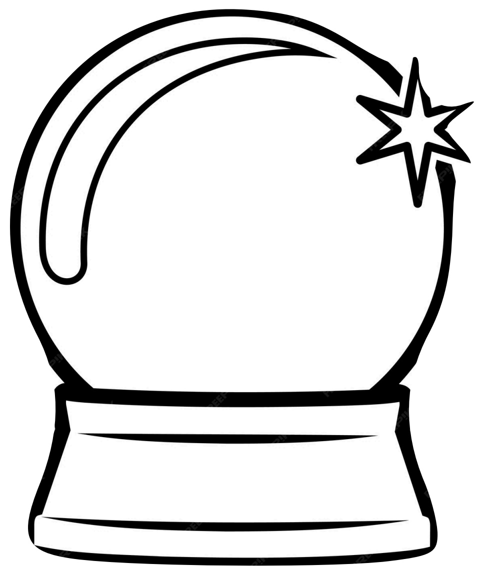
page 1	page 2	page 3	
6 7 9 11 2	8 9 6 3 11	7 10 4 1 5	ooo

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

Brandeis UNIVERSITY

Let's **think** together

Understanding the implications of access granularity



query: x<4

120 B (3 I/O)

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

oracle

oracle

size of result set = 24 B

7 10 4 1 5	8 9 6 3 11	2 3 1
		output buffer

memory
(size = 120 B)

SSD
(size = few GBs)

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

Brandeis UNIVERSITY

Was the index **helpful**?

Let's **think** together

Understanding the implications of access granularity

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

Thought Experiment 4

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

Thought Experiment 5

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

memory
(size = 120 B)

SSD
(size = few GBs)

page 1	page 2	page 3	
6 7 9 11 2	8 9 6 3 11	7 10 4 1 5	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

Brandeis UNIVERSITY

Project 1

Testing the waters!

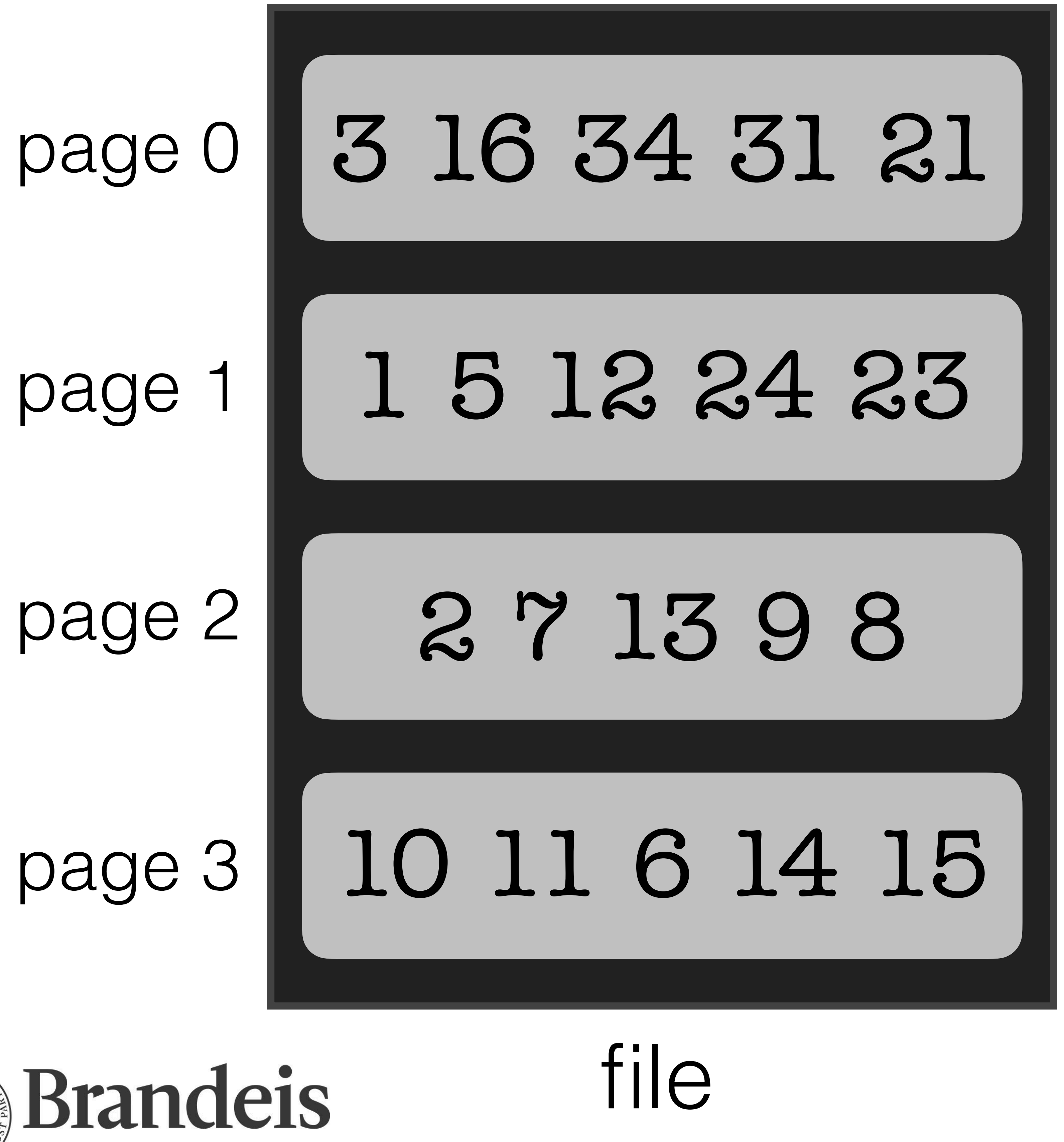
Implementing a Simple Zone Map

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)

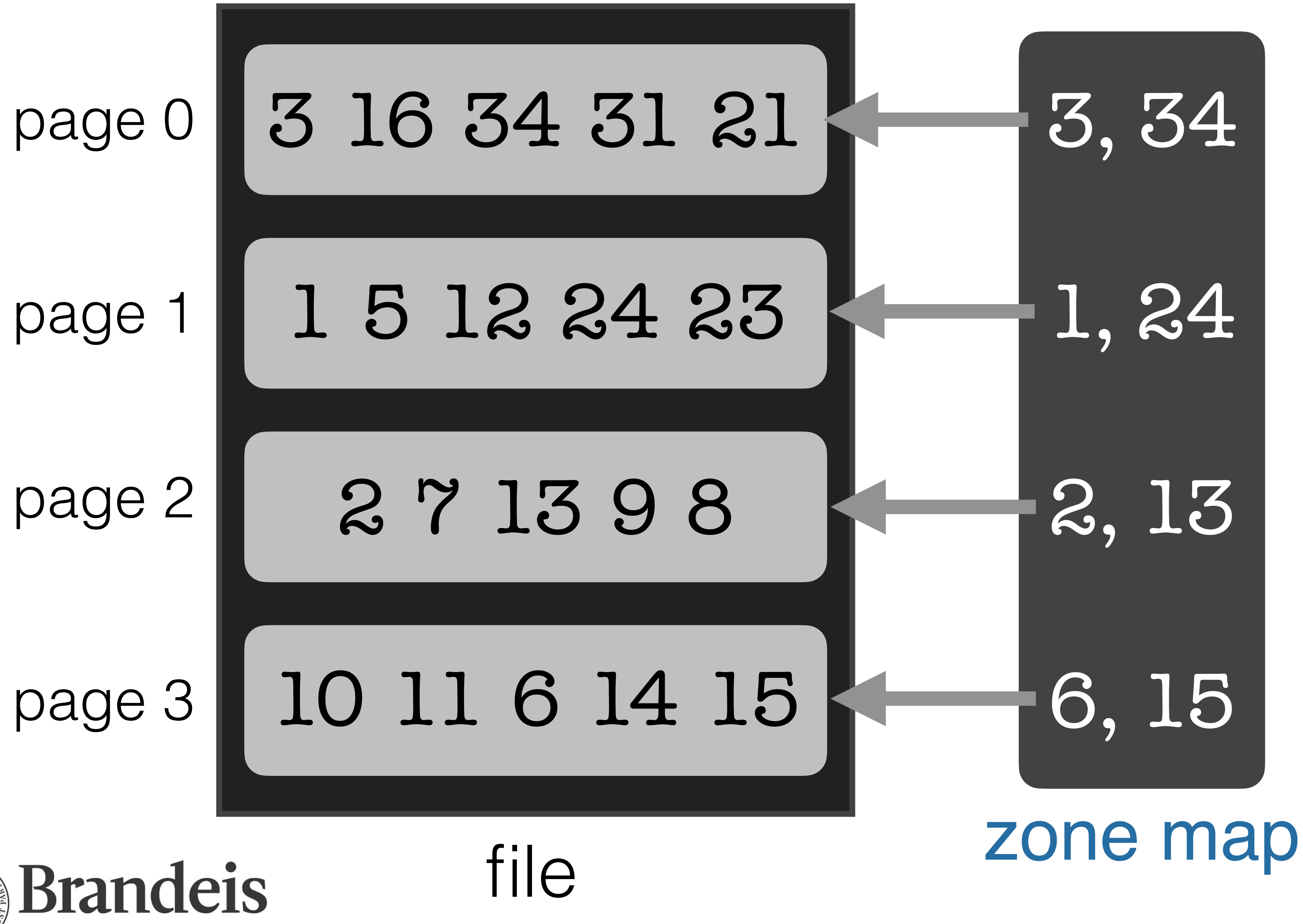
a file is a collection of **pages** (blocks)

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

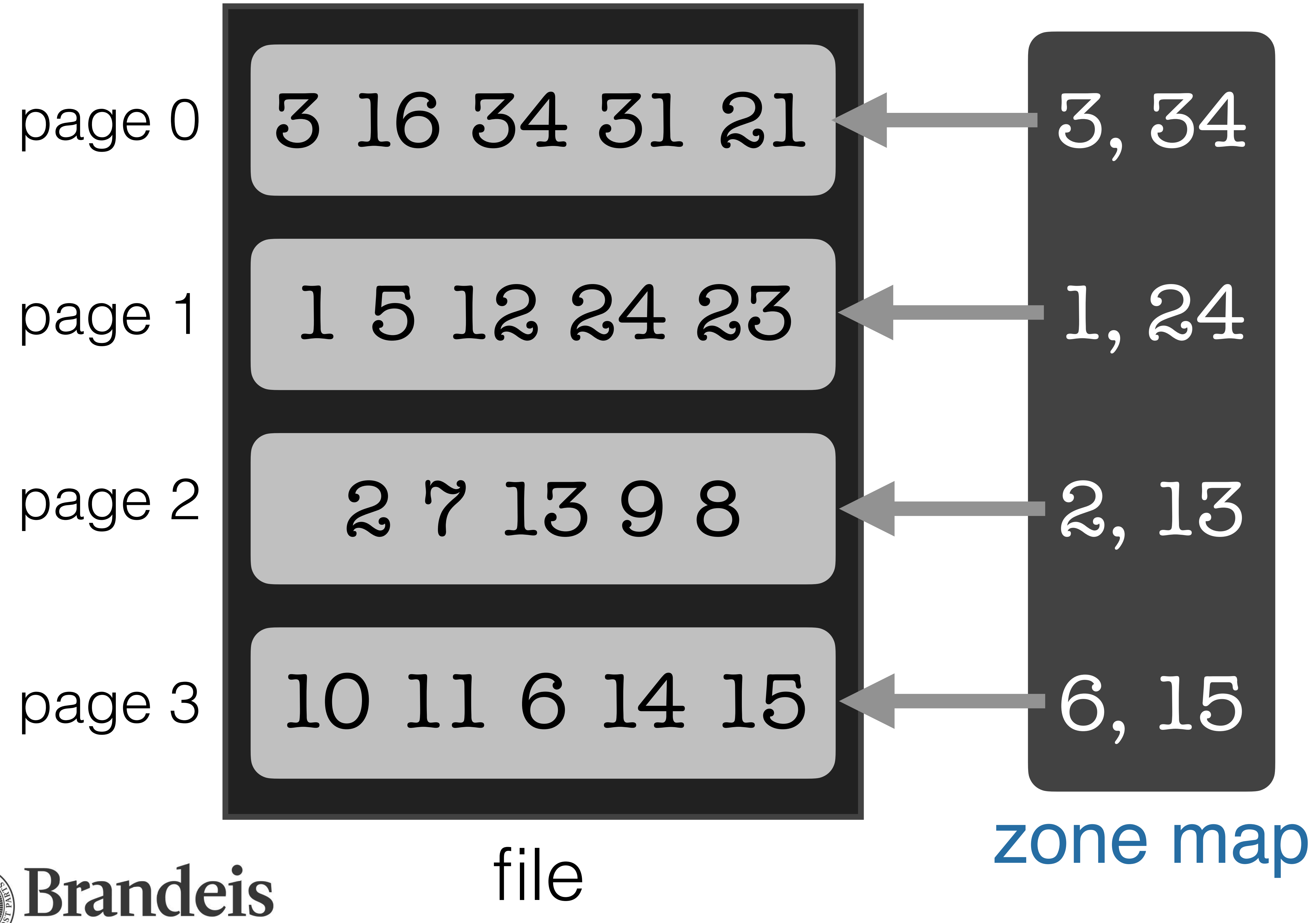
x=20: 4 I/O

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

Thought Experiment 6

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

Readings for next class

Papers, papers, and papers

<u>The Seattle Report on Database Research</u>
— J. Hellerstein, M. Stonebraker and J. Hamilton
SIGMOD Record, 2022

<u>The Seattle Report on Database Research</u>
— 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

Brandeis
UNIVERSITY

Assignment submission

And grading!

gradescope

Code: **PYG88X**

don't be late in your submissions!

Next time in COSI 167A

Introduction to column stores

fundamentals of data **storage**

introduction to **row-stores** and **column-stores**

COSI 167A
Advanced Data Systems

Class 2

Data Systems 101

Prof. Subhadeep Sarkar

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

Brandeis University