

CS 7270: Advanced Database Systems Fall 2025

# Lecture 02

## Data system architecture

Prashant Pandey

[p.pandey@northeastern.edu](mailto:p.pandey@northeastern.edu)

Some reminders...

no  
smartphones



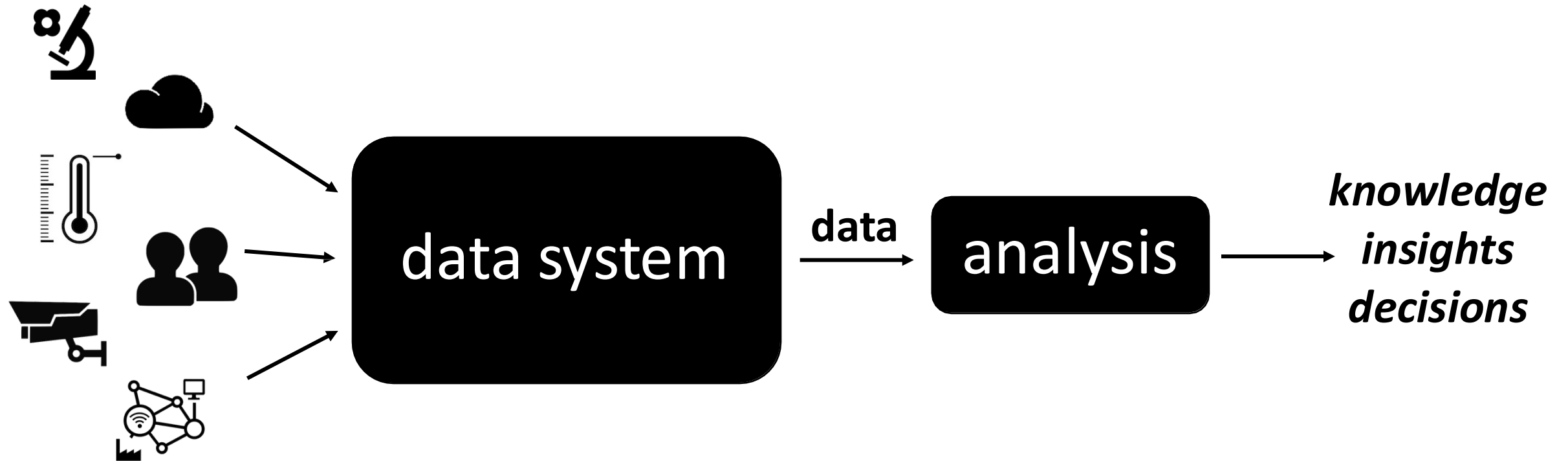
no  
laptop



# Data system architecture essentials

Acknowledgement: Slides taken from Prof. Manos Athanassoulis, BU

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



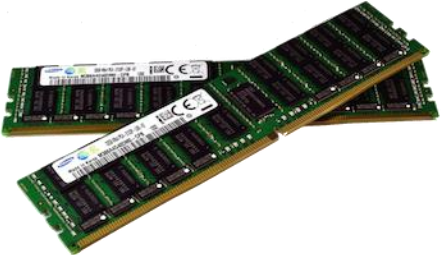
# Growing need for tailored systems



new applications



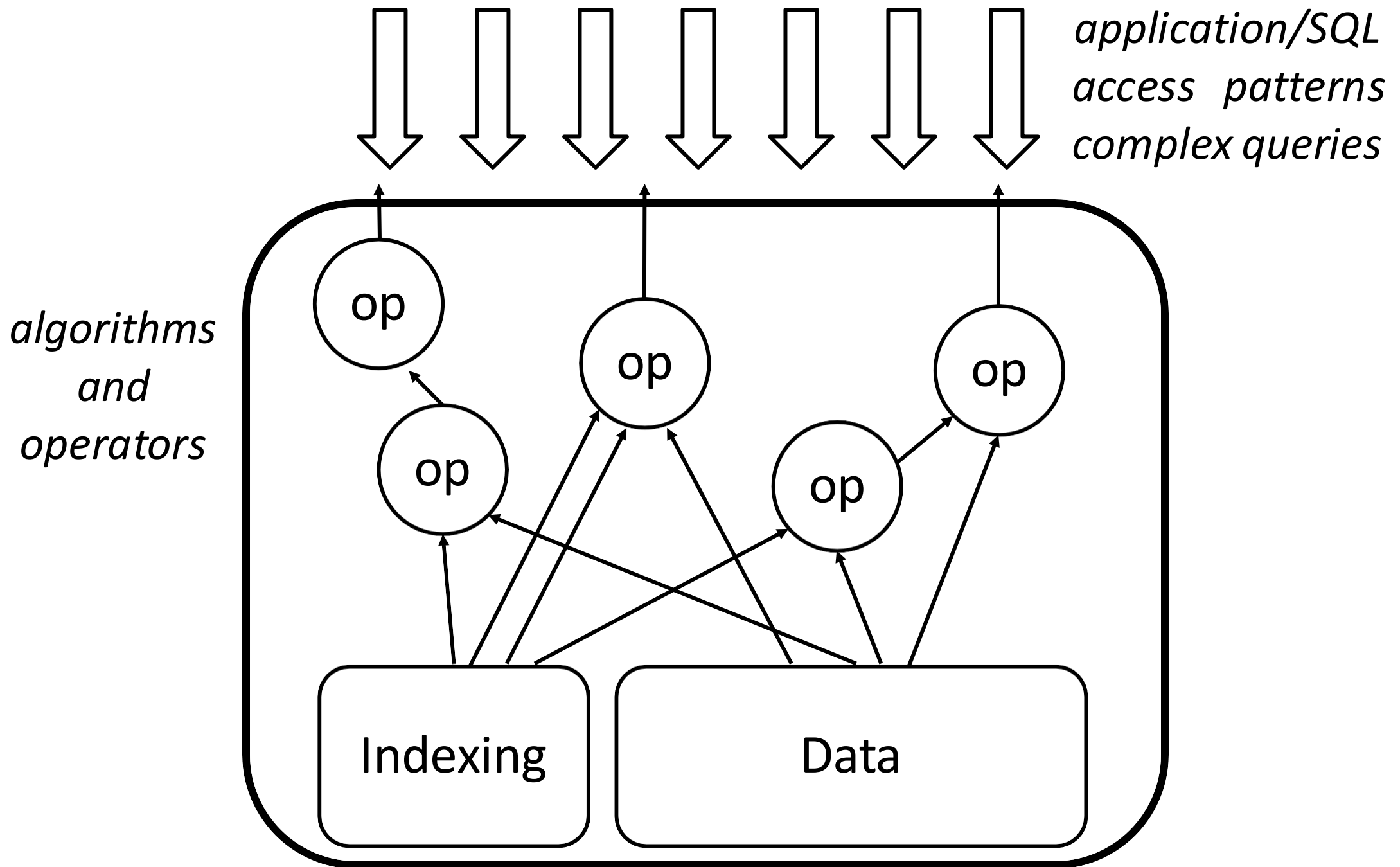
new hardware

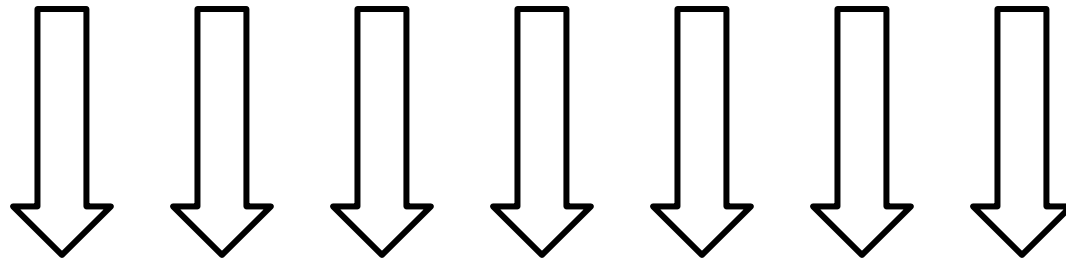


more data



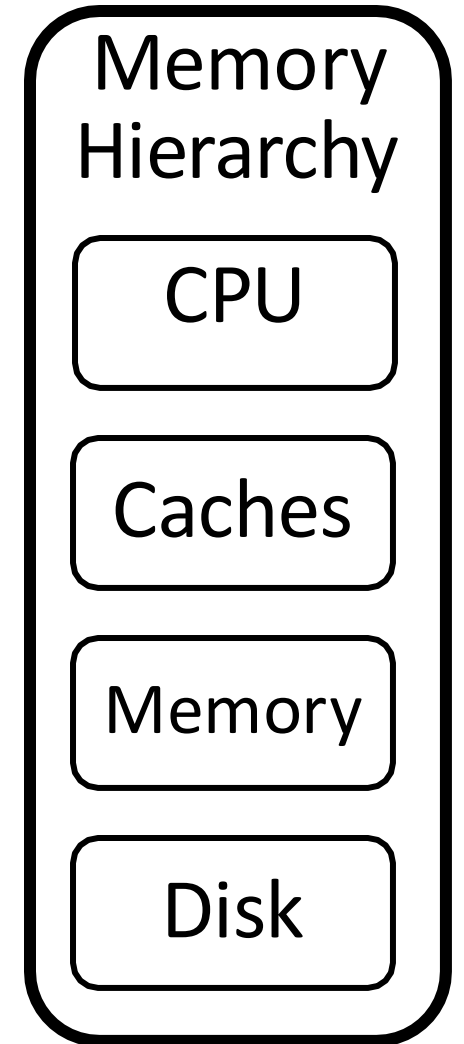
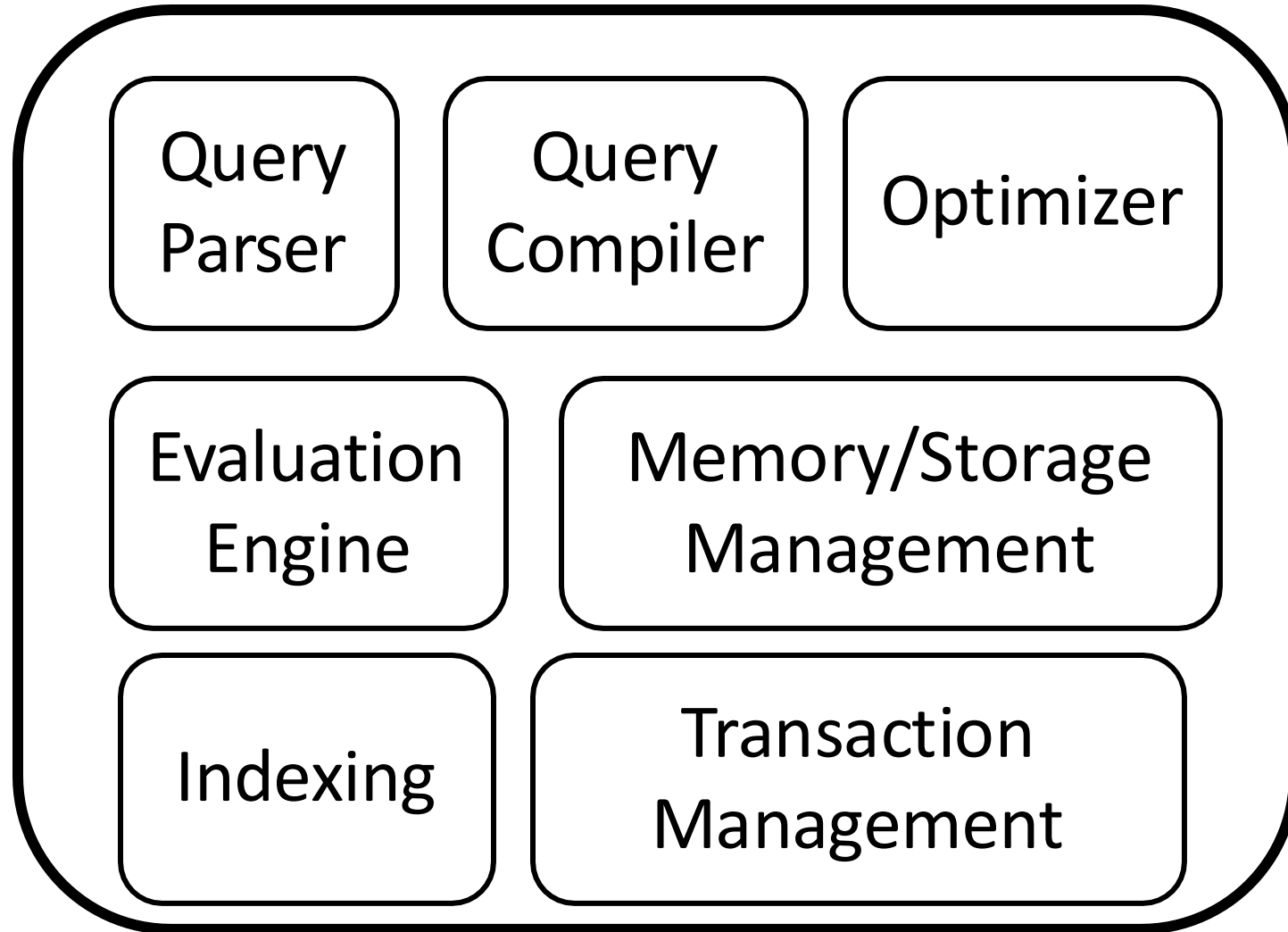
Data system, what's inside?





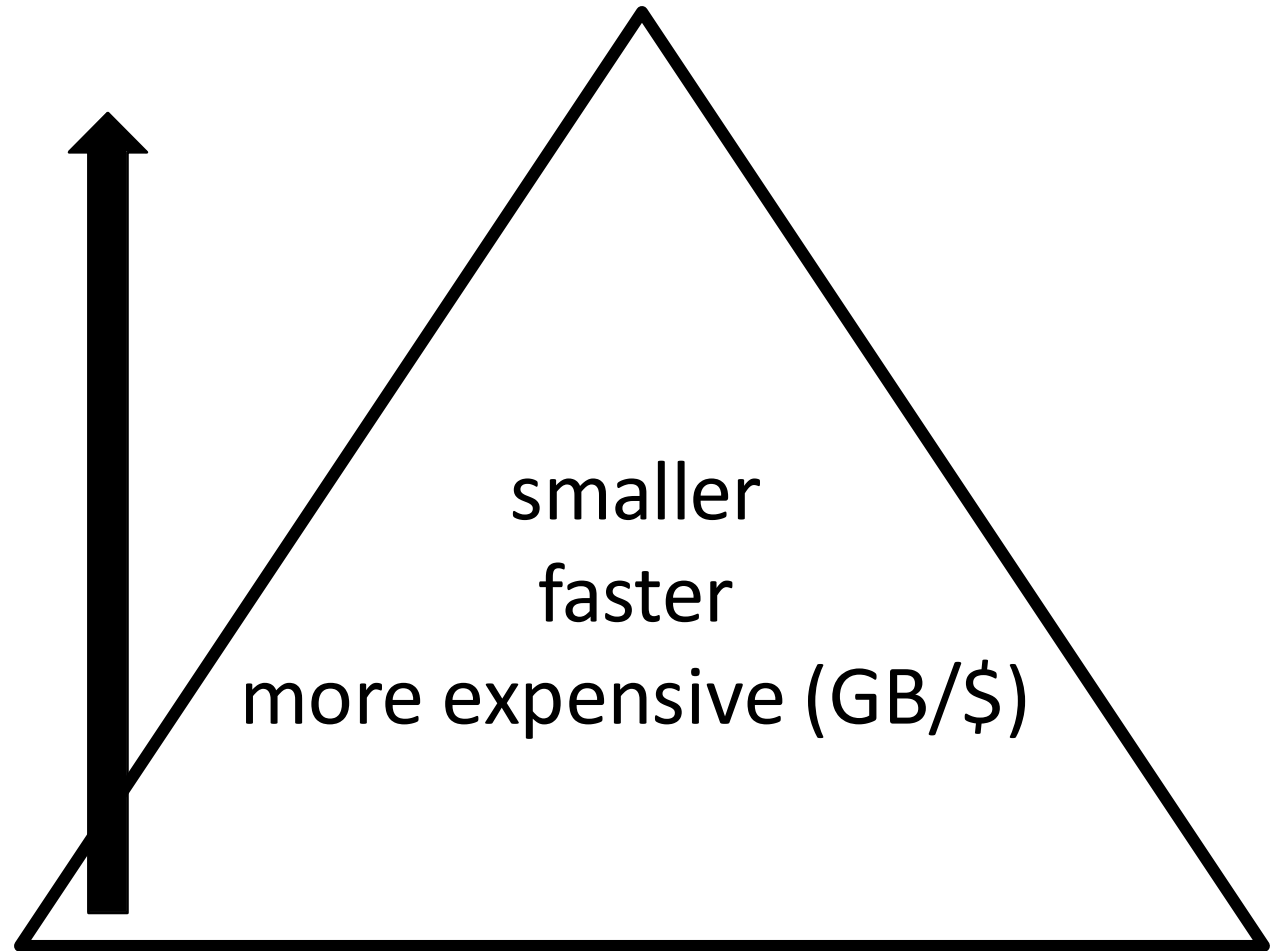
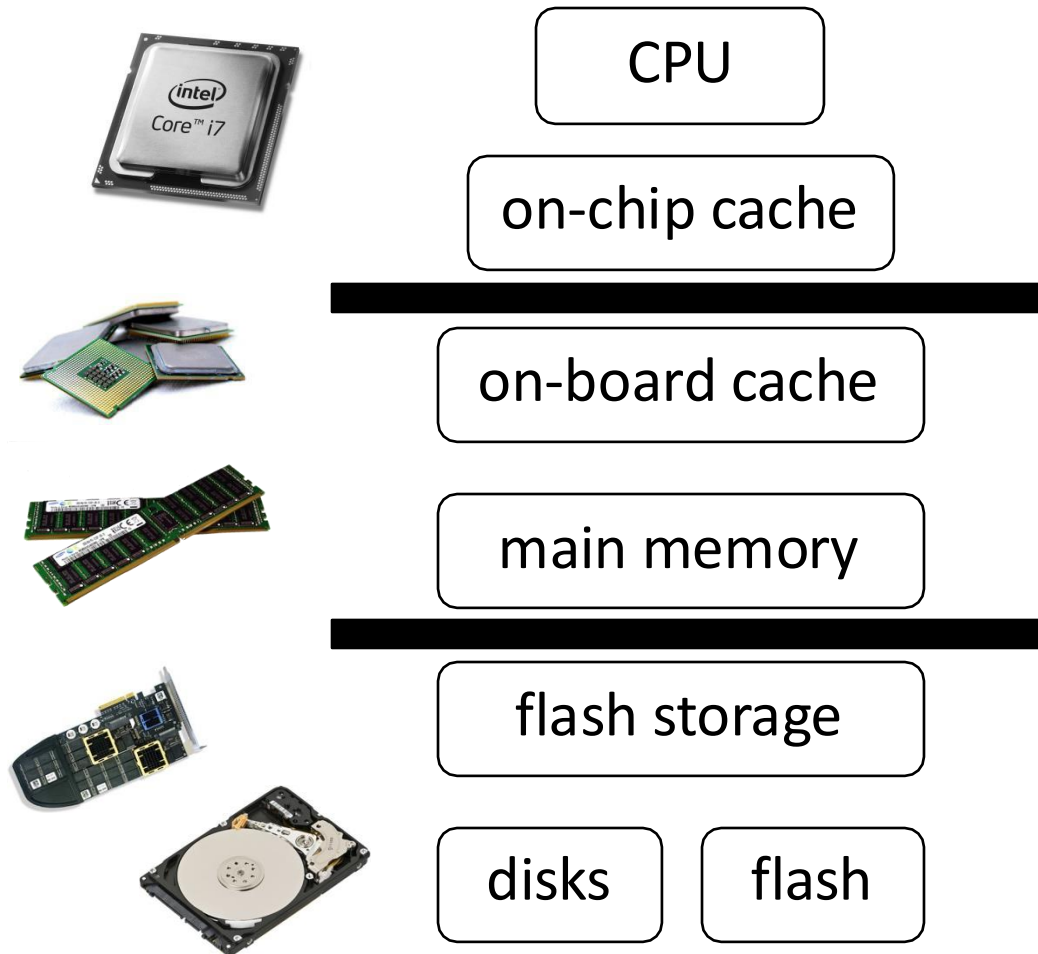
*application/SQL  
access patterns  
complex queries*

*modules*

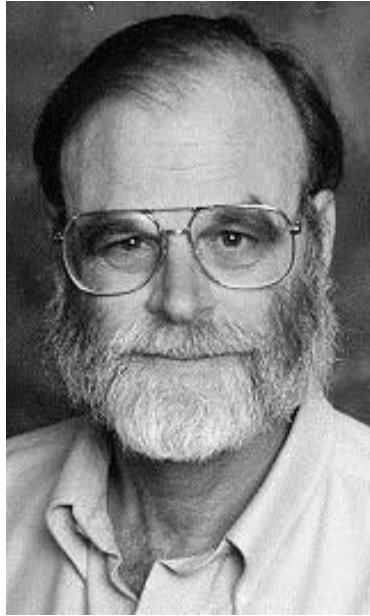


Data system, what's underneath?

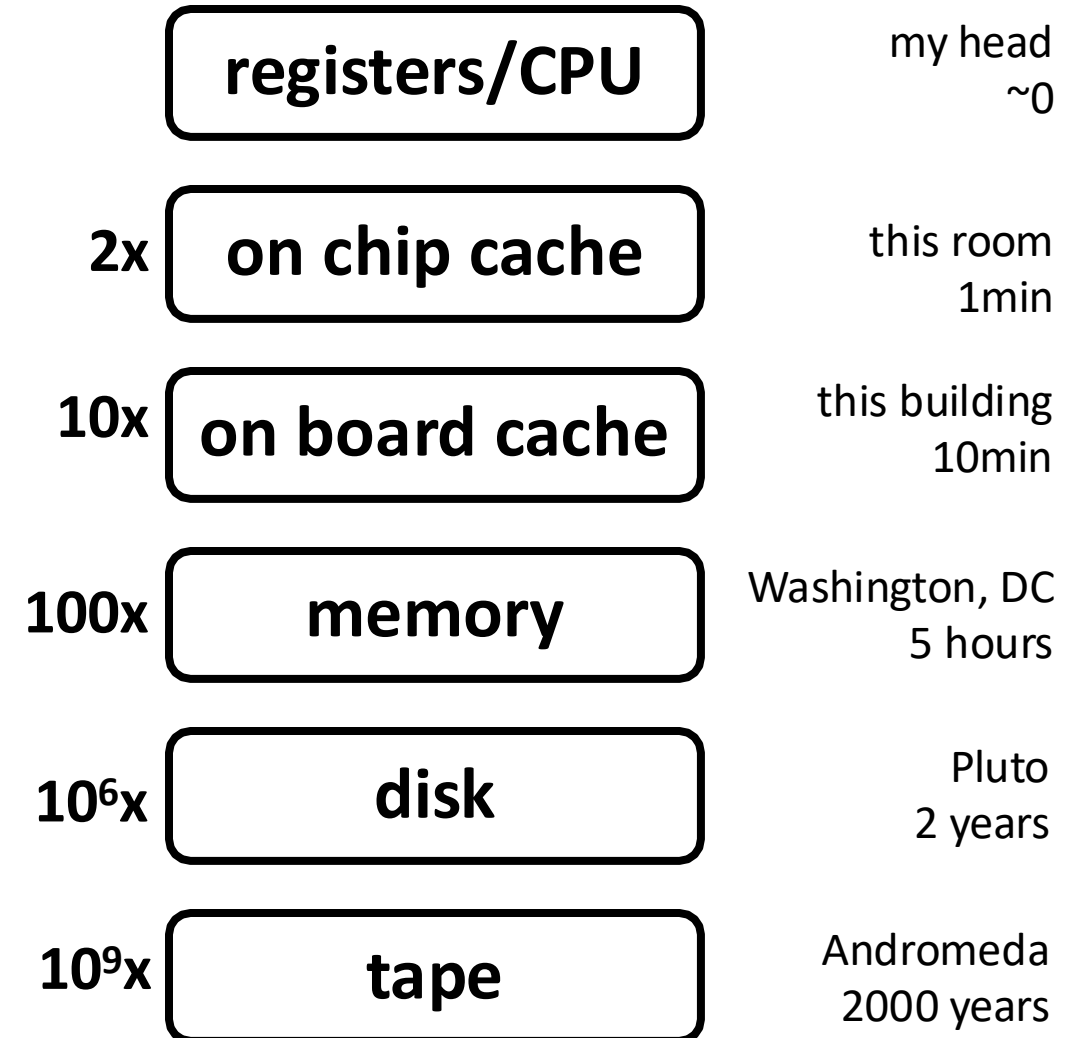
# Memory hierarchy



# Memory hierarchy (by Jim Gray)



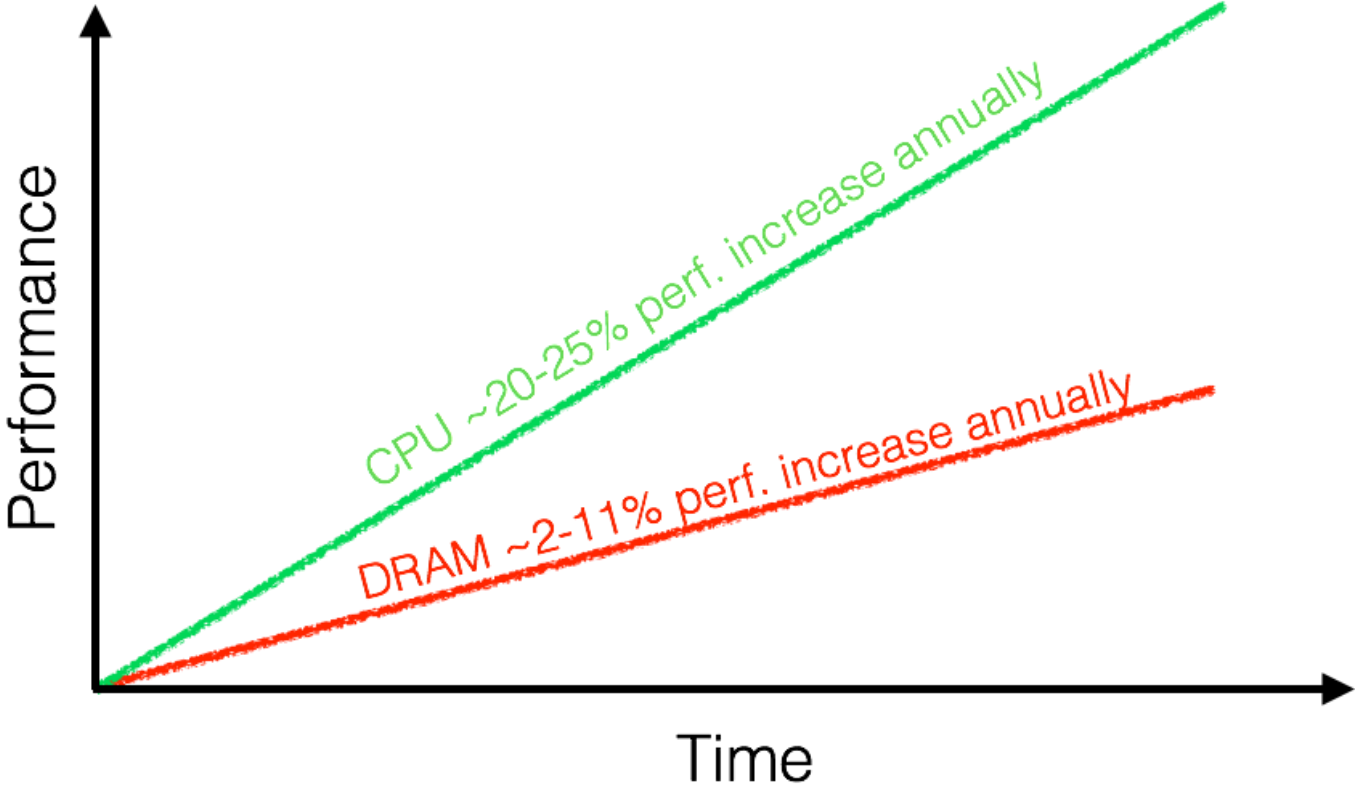
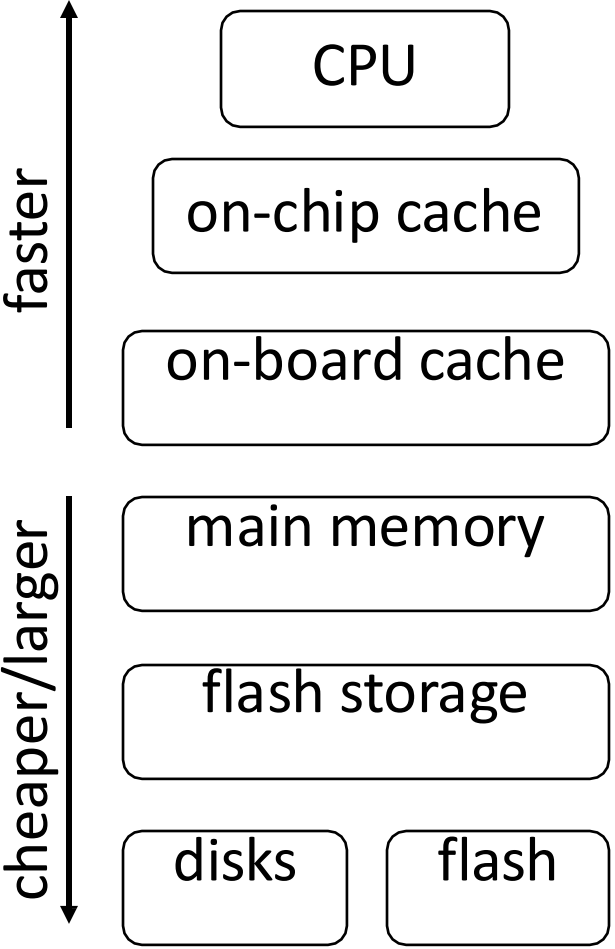
Jim Gray, IBM, Tandem, Microsoft, DEC  
“The Fourth Paradigm” is based on his vision  
**ACM Turing Award 1998**  
**ACM SIGMOD Edgar F. Codd Innovations award 1993**



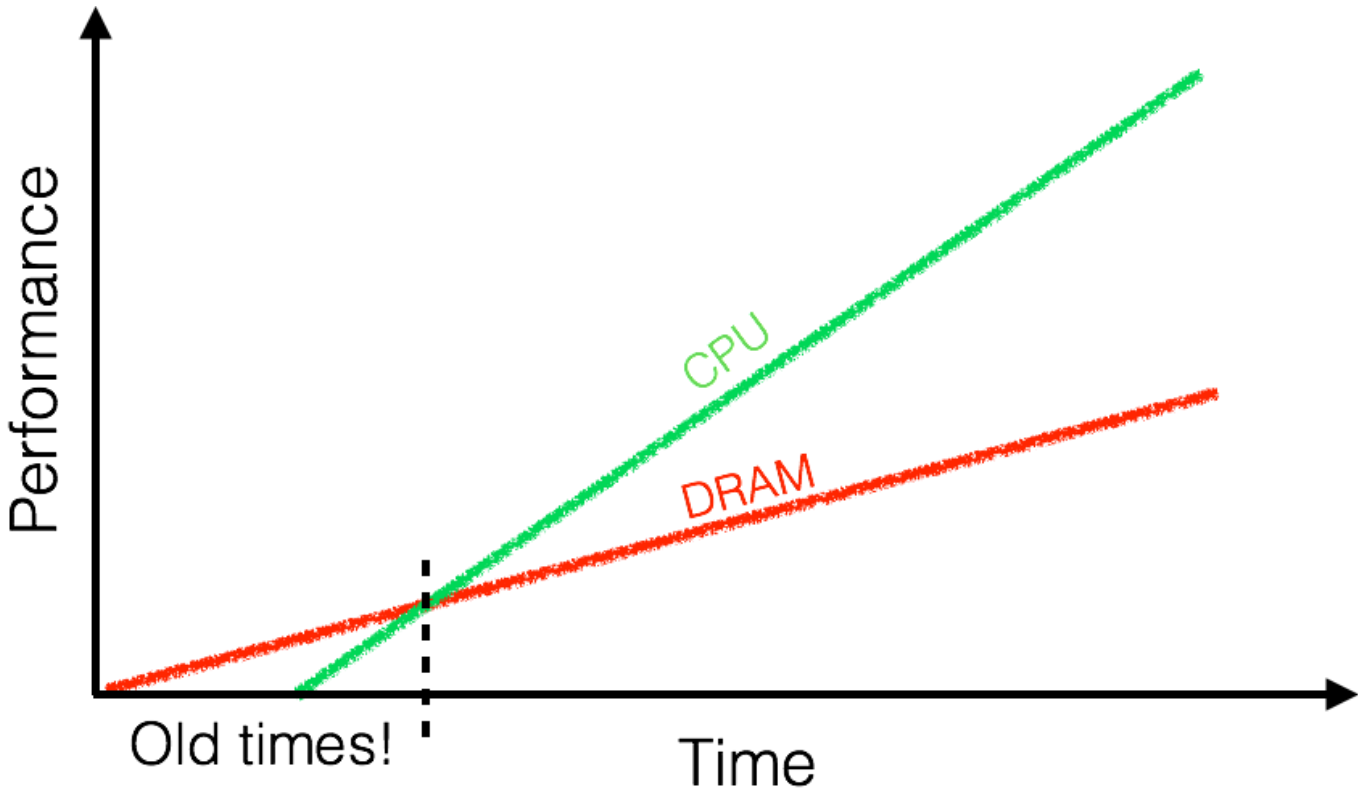
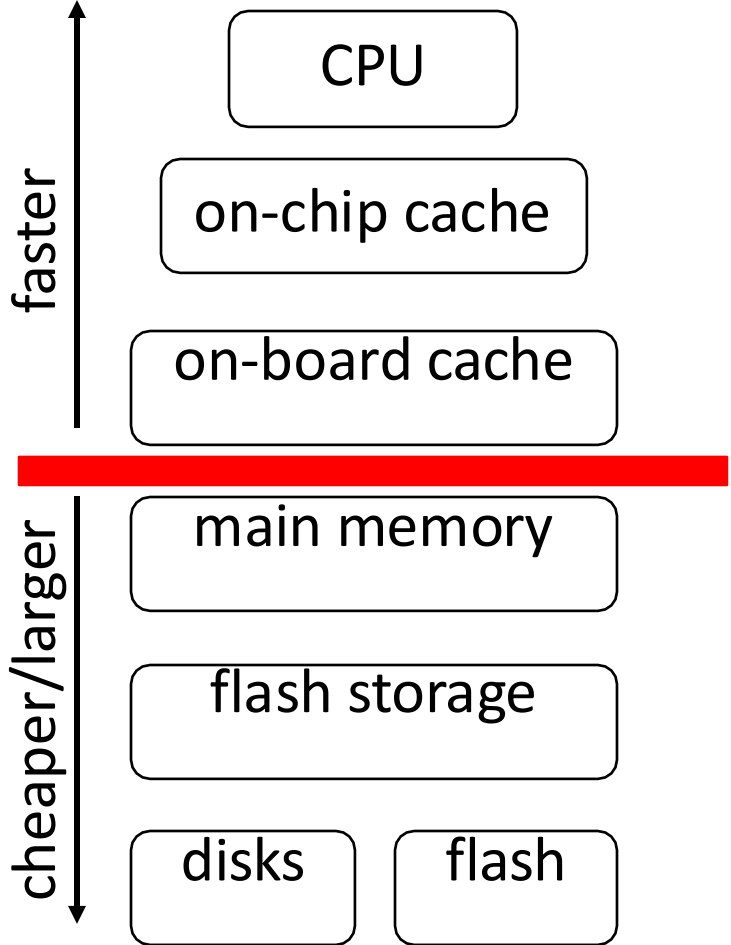
# Memory hierarchy and latencies

Memory	Size	Latency	Bandwidth
L1 cache	32 KB	1 nano sec	1 TB/sec
L2 cache	256 KB	4 nano sec	1 TB/sec (shared by cores)
L3 cache	8 MB or more	~30-40 nano sec	> 400 GB/sec
Main memory DDR DIMM	4 GB to 1 TB	~80-100 nano sec	100 GB/sec
I/O devices on memory bus	6 TB	100X-1000X slower than memory	25 GB/sec
I/O devices on PCIe bus	Limited only by cost	Milli sec – minutes	GB-TB/hour (depends on HW and distance)

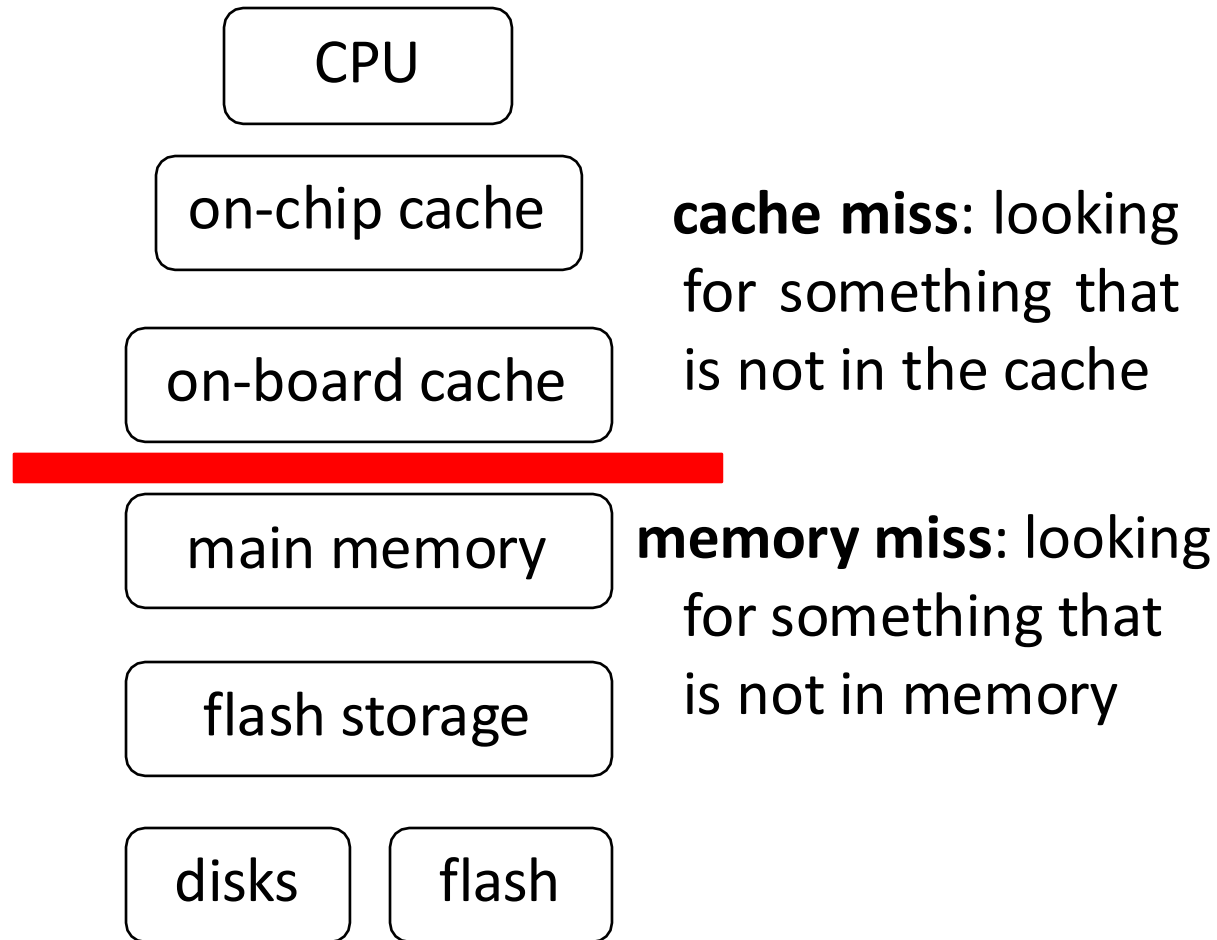
# Memory wall



# Memory wall



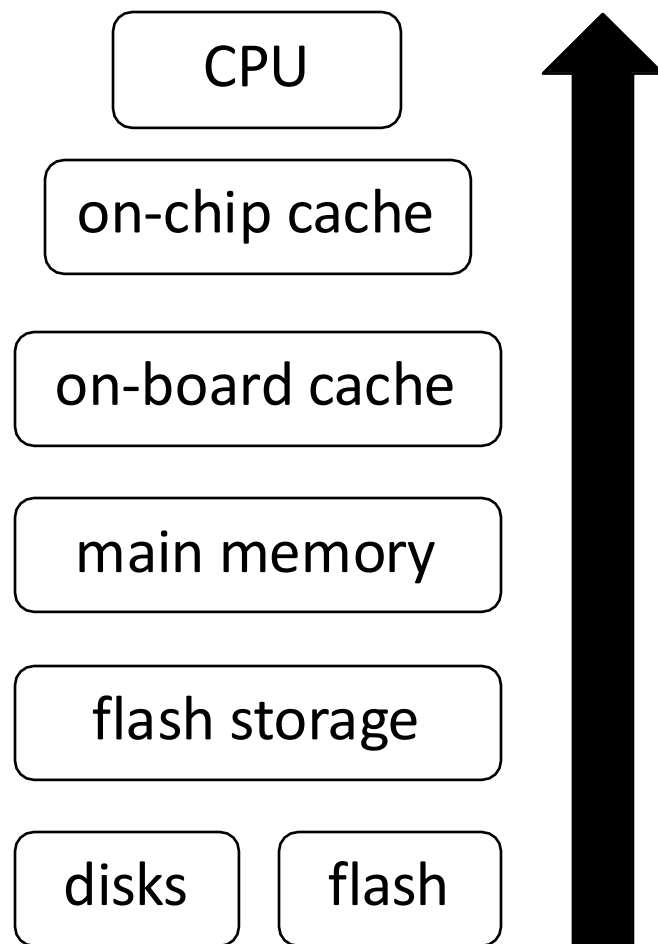
# Cache/memory misses



**what happens if I miss?**



# Data movement



data go through  
all necessary levels

also read  
*unnecessary* data

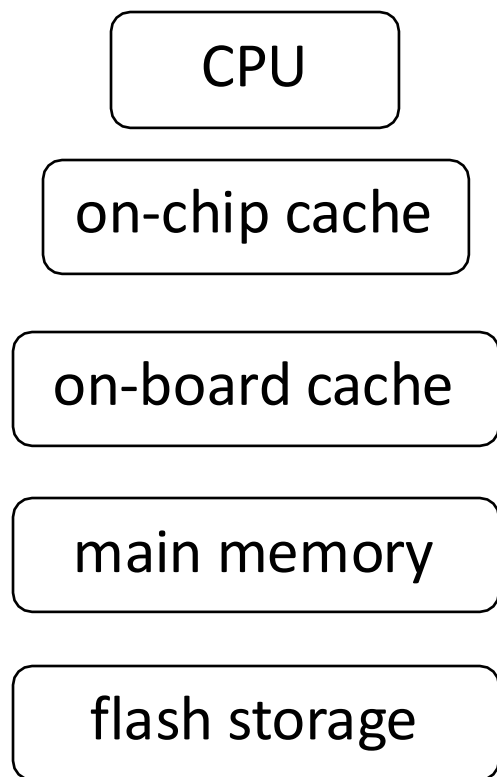


Photo by Gary Dineen/NBAE via Getty Images

need to read only X  
read the whole page



# Data movement



data go through  
all necessary levels

also read  
*unnecessary* data

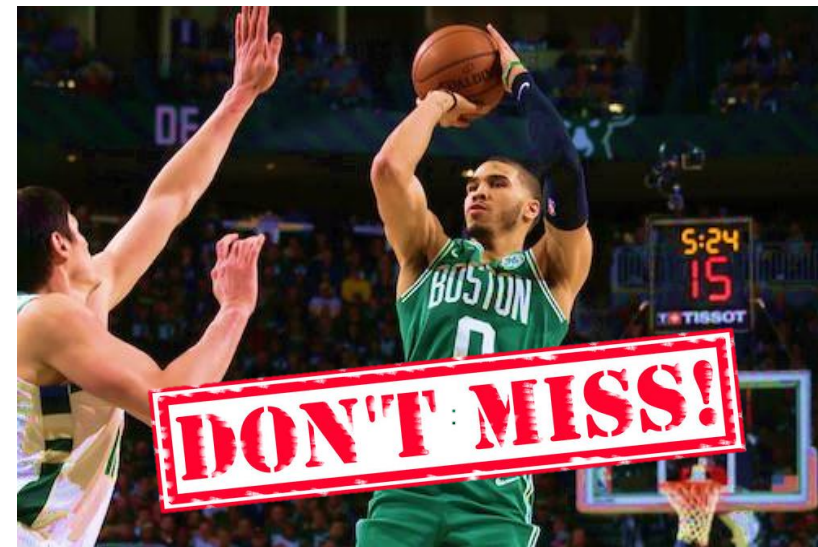


Photo by Gary Dineen/NBAE via Getty Images

need to read only X  
read the whole page



remember!

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

# Page-based access & random access

query  $x < 7$

scan



size=120 bytes

**memory (memory level N)**

**disk (memory level N+1)**

1, 5, 12, 24, 23

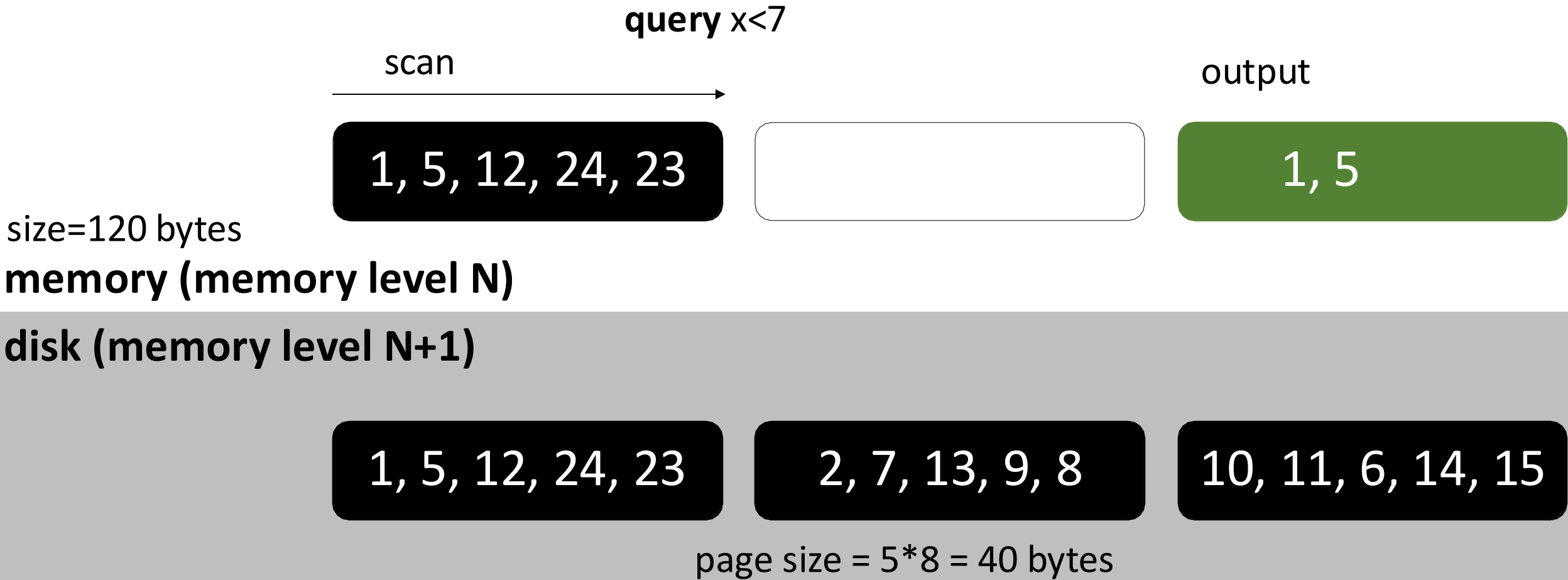
2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size =  $5 * 8 = 40$  bytes

\$ 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

\$ 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

\$ 80 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

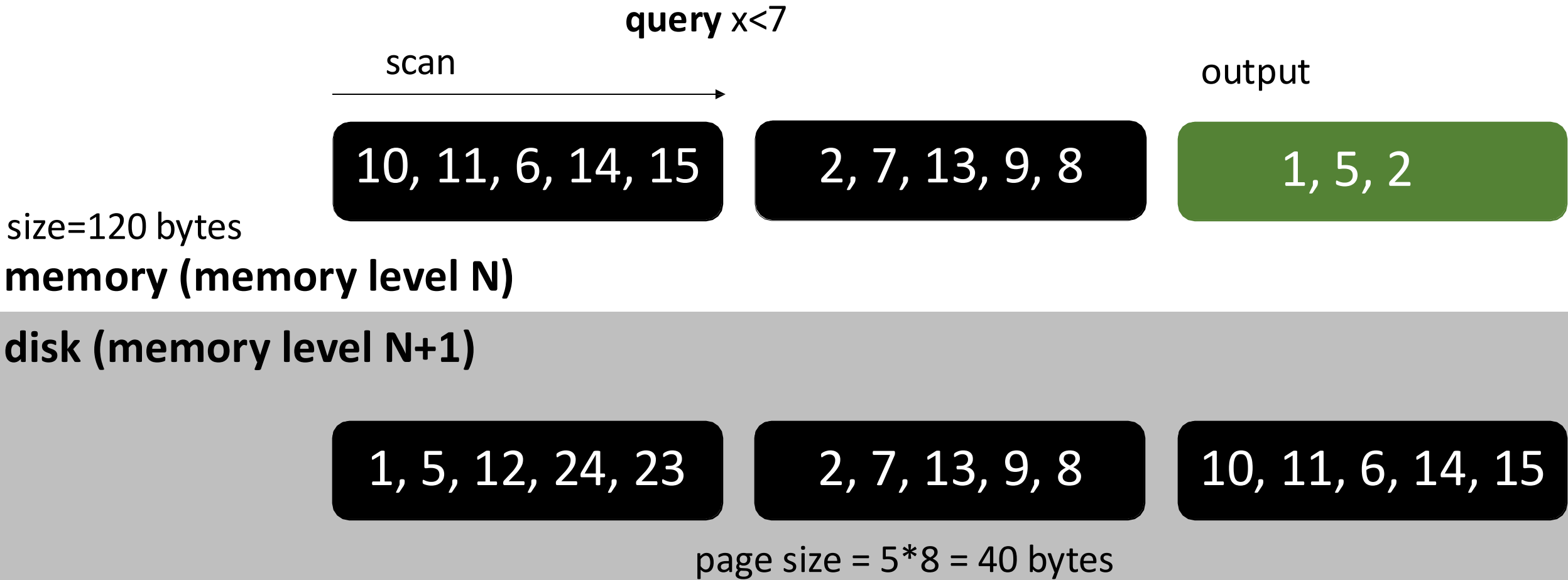
2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size =  $5 * 8 = 40$  bytes

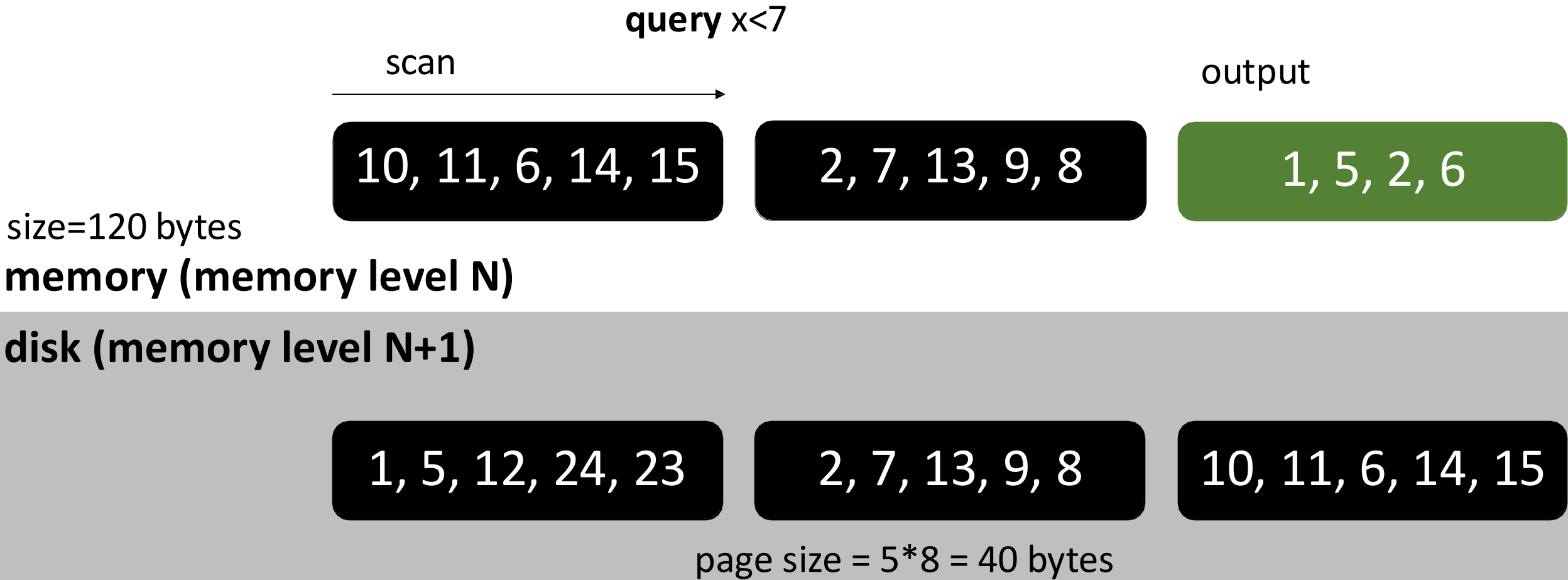
\$ 80 bytes

# Page-based access & random access



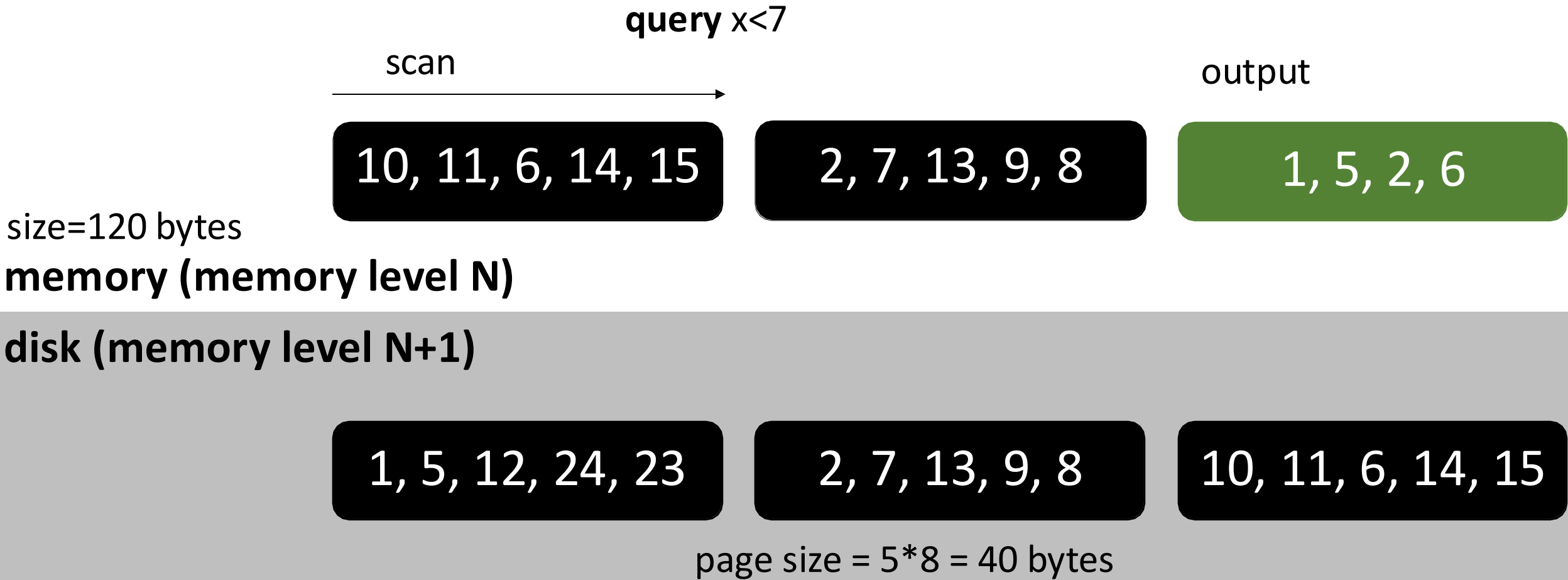
\$ 80 bytes

# Page-based access & random access



\$ 120 bytes

# Page-based access & random access



What if we had an oracle (perfect index)?



# Page-based access & random access

query  $x < 7$

scan



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

\$ 40 bytes

# Page-based access & random access

query  $x < 7$

oracle

1, 5, 12, 24, 23

output

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

\$ 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

\$ 40 bytes

# Page-based access & random access

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

\$ 80 bytes

# Page-based access & random access

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

\$ 80 bytes

# Page-based access & random access

query  $x < 7$

oracle

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

\$ 80 bytes

# Page-based access & random access

query  $x < 7$

oracle

output

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

# Page-based access & random access

\$ 120 bytes



query  $x < 7$

oracle

*was the oracle helpful?*

output

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?

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

how we store data

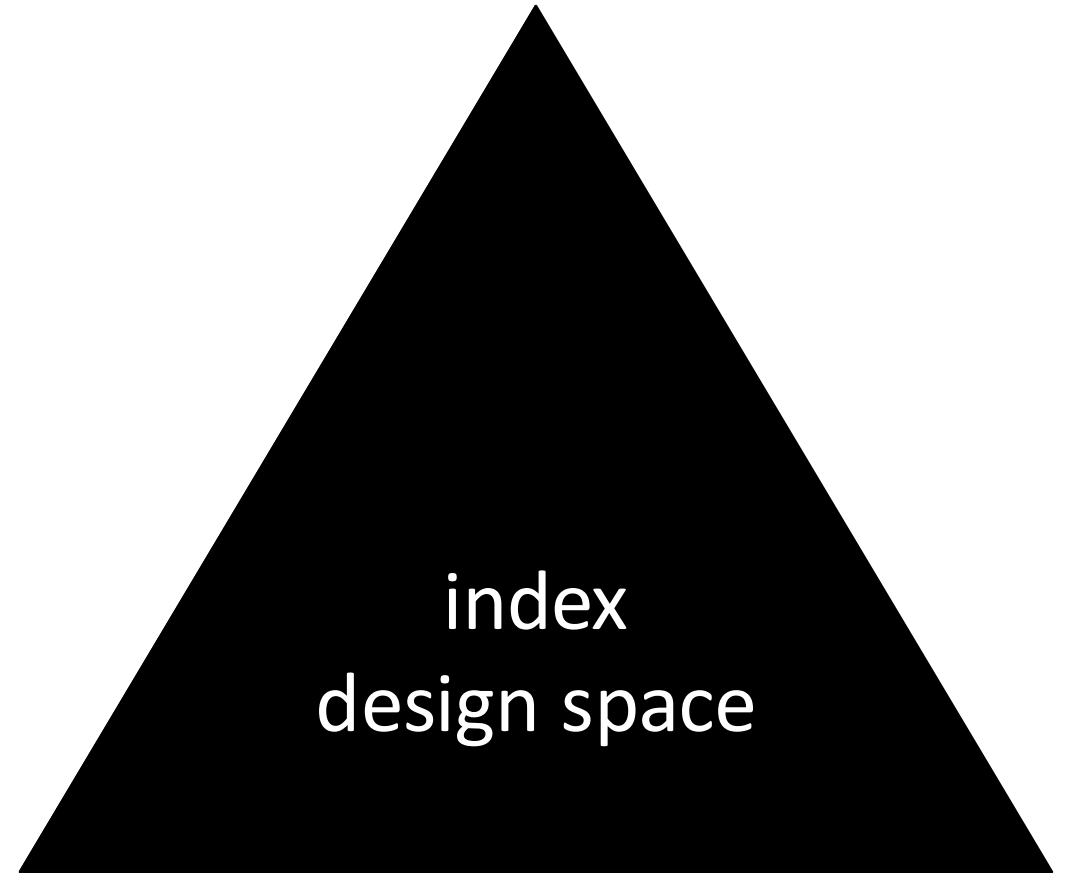
layouts, indexes

every **byte** counts

overheads and tradeoffs

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);

read random next;



ideal random access?

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

# The language of efficient systems: C/C++

*why?*

low-level control over hardware

make decisions about physical data placement and consumptions

fewer assumptions

# The language of efficient systems: C/C++

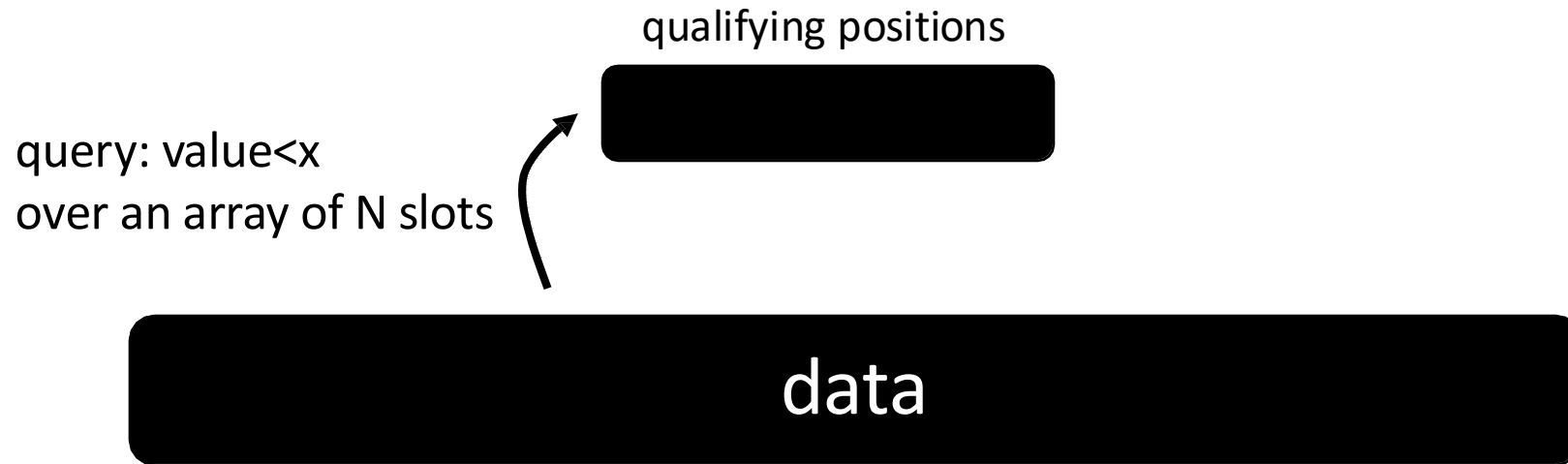
*why?*

low-level control over hardware

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

# A “simple” database operator

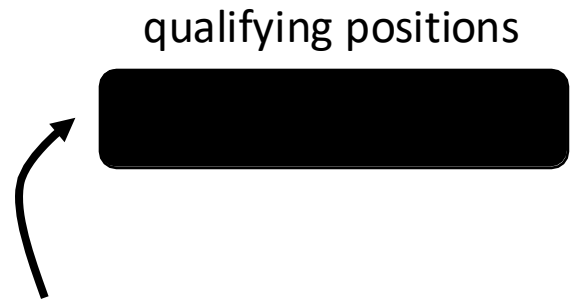
*select operator (scan)*





How to implement it?

```
result = new array[data.size];  
j=0;  
for (i=0; i<data.size; i++)  
  if (data[i]<x)  
    result[j++]=i;
```



query: value<x  
over an array of N slots

what if only 0.1% qualifies?

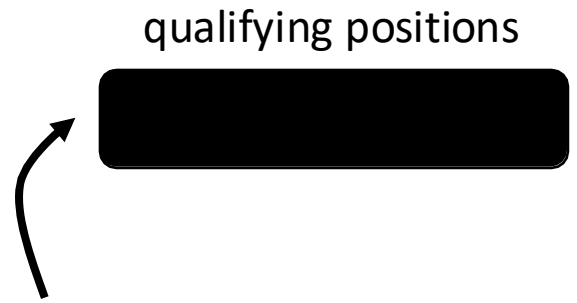
memory





How to implement it?

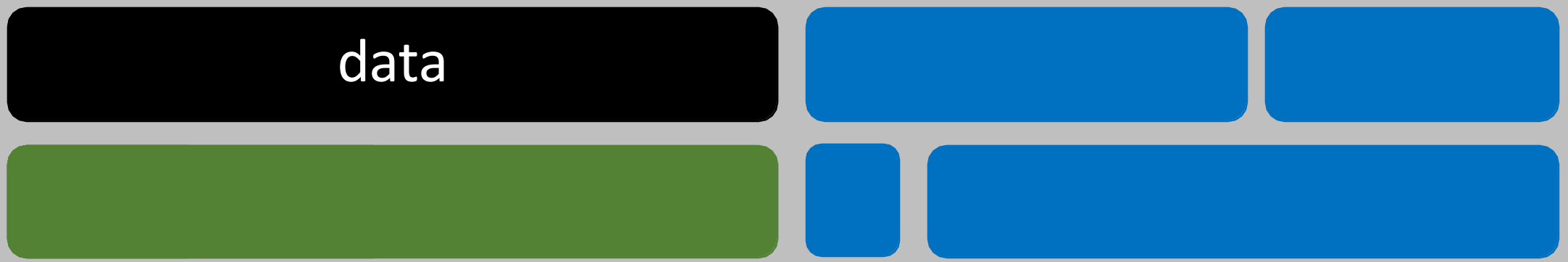
```
result = new array[data.size];  
j=0;  
for (i=0; i<data.size; i++)  
  if (data[i]<x)  
    result[j++]=i;
```



query: value<x  
over an array of N slots

what if only 0.1% qualifies?

memory



```
result = new array[data.size];  
j=0;  
for (i=0; i<data.size; i++)  
  if (data[i]<x)  
    result[j++]=i;
```

needs coordination!  
what about result writing?

qualifying positions



query: value<x  
over an array of N slots



what about multi-core?  
NUMA? SIMD? GPU?

data

core1

core2

core3

core4



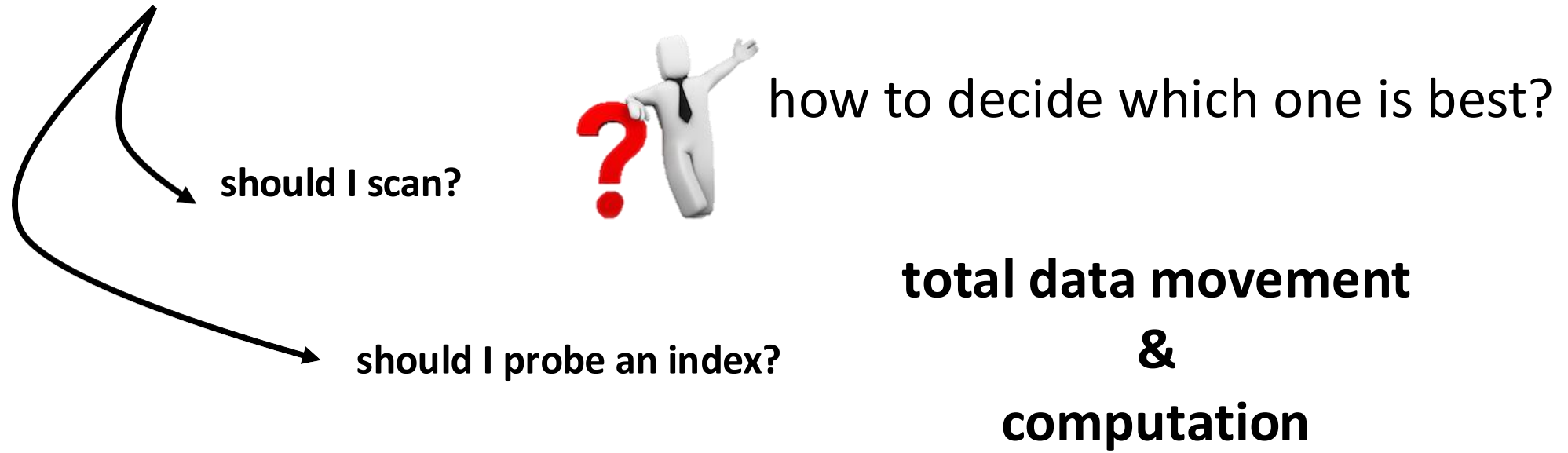
What about having multiple queries?

```
result = new array[data.size];  
j=0;  
for (i=0; i<data.size; i++)  
  if (data[i]<x)  
    result[j++]=i;
```



query1: value<x1  
query2: value<x2 ...





# Next class

- In-memory indexing

***Make sure to read the related papers from the reading list***