CS 591: Data Systems Architectures

class 2

Data Systems 101

Prof. Manos Athanassoulis

https://midas.bu.edu/classes/CS591A1/
some reminders

no smartphones

no laptop
class summary

2 classes per week / OH 5 days per week

each student
1 presentation/discussion lead + 2 reviews/questions per week

systems or research project + proposal + mid-semester report
systems project
implementation-heavy C/C++ project
groups of 2

research project
groups of 3
pick a subject (list will be available)
design & analysis
experimentation
Week 2 – register for presentations by 1/30
first presentation on 2/4

Week 3 – form groups by 2/7

Week 4 – find project by 2/14

Week 5 – submit project proposal on 2/21

Week 9 – submit mid-semester project report on 3/21

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

**discussions interaction in OH questions**
Piazza

all discussions & announcements

http://piazza.com/bu/spring2020/cs591a1/

also available on class website

10 already registered!

register so we can reach you easily
big data
(it’s not only about size)

The 3 V’s

size (volume)
rate (velocity)
sources (variety)

+ our ability to collect *machine-generated* data

scíentíficó experiments

social

sensos

Internet-of-Things
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.

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

**data system**

some analysis

**data** → **analysis** → **knowledge insights decisions**
data system, what’s inside?
application/SQL
access patterns
complex queries

Indexing

Data

algorithms and operators
application/SQL access patterns complex queries
growing environment

db
large systems
complex
lots of tuning
legacy

noSQL
simple, clean
“just enough”

newSQL

need for scalability

more complex
applications

>$200B by 2020, growing at 11.7% every year
[The Forbes, 2016]

$3B by 2020, growing at 20% every year
[Forrester, 2016]
growing need for tailored systems

new applications

new hardware

more data
data system, what’s underneath?
memory hierarchy

CPU
- on-chip cache
- on-board cache
- main memory
- flash storage
- disks
- flash

smaller faster
more expensive (GB/$)
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

<table>
<thead>
<tr>
<th>Level</th>
<th>Speed Factor</th>
<th>Storage Type</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>registers/CPU</td>
<td>2x</td>
<td>on chip cache</td>
<td>my head ~0</td>
</tr>
<tr>
<td>on board cache</td>
<td>10x</td>
<td></td>
<td>this room 1min</td>
</tr>
<tr>
<td>memory</td>
<td>100x</td>
<td></td>
<td>this building 10min</td>
</tr>
<tr>
<td>disk</td>
<td>10^6x</td>
<td></td>
<td>Washington, DC 5 hours</td>
</tr>
<tr>
<td>tape</td>
<td>10^9x</td>
<td></td>
<td>Pluto 2 years</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Andromeda 2000 years</td>
</tr>
</tbody>
</table>
memory hierarchy (by Jim Gray)

- registers/CPU
- on chip cache (2x)
- on board cache (10x)

Tape? sequential-only magnetic storage still a multi-billion industry
Jim Gray (a great scientist and engineer)

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

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

- CPU
- on-chip cache
- on-board cache
- main memory
- flash storage
- disks
- flash

Performance vs. Time

- CPU ~20-25% perf. increase annually
- DRAM ~2-11% perf. increase annually
memory wall

- CPU
- on-chip cache
- on-board cache
- main memory
- flash storage
- disks
- flash

Performance vs. Time

Old times!

Faster and cheaper/larger memory options compared to traditional storage technologies.
cache/memory misses

- CPU
  - on-chip cache
  - on-board cache
  - main memory
  - flash storage
  - disks
  - flash

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

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

what happens if I miss?
data movement

- CPU
- on-chip cache
- on-board cache
- main memory
- flash storage
- disks

Data go through all necessary levels.

Also read unnecessary data.

Need to read only X page.

X page
Data movement:

- CPU
- On-chip cache
- On-board cache
- Main memory
- Flash storage

Data go through all necessary levels. Also read unnecessary data. Need to read only X page.

Remember! Disk is millions (mem, hundreds) 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 = $5 \times 8 = 40$ bytes
page-based access & random access

memory (memory level N)

| 1, 5, 12, 24, 23 |

query $x < 7$

output

| 1, 5 |

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

size=120 bytes

$40$ bytes
page-based access & random access

memory (memory level N)

size=120 bytes

query \( x < 7 \)

scan

output

\[ 1, 5, 12, 24, 23 \]

\[ 2, 7, 13, 9, 8 \]

\[ 1, 5 \]

 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 $x < 7$

\[
\begin{array}{c}
1, 5, 12, 24, 23 \\
2, 7, 13, 9, 8 \\
10, 11, 6, 14, 15
\end{array}
\]

size = 120 bytes

memory (memory level N)

disk (memory level N+1)

page size = 5*8 = 40 bytes

output $1, 5, 2$
page-based access & random access

memory (memory level $N$)

size = 120 bytes

query $x < 7$

scan

output

1, 5, 2

disk (memory level $N+1$)

page size = $5 \times 8 = 40$ bytes

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

$80 \text{ bytes}$
page-based access & random access

memory (memory level N)

size=120 bytes

disk (memory level N+1)

size=120 bytes

query $x < 7$

scan

output

$\begin{align*}
\text{scan} & : [10, 11, 6, 14, 15] \\
\text{query} & : [2, 7, 13, 9, 8] \\
\text{output} & : [1, 5, 2]
\end{align*}$

page size = 5*8 = 40 bytes

$\text{size=120 bytes}$

$\text{page size = 5*8 = 40 bytes}$
page-based access & random access

memory (memory level N)

size = 120 bytes

query \( x < 7 \)

scan

output

10, 11, 6, 14, 15
2, 7, 13, 9, 8
1, 5, 2, 6

disk (memory level N+1)

page size = 5*8 = 40 bytes

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

$80$ bytes
page-based access & random access

size=120 bytes

count

memory (memory level N)

disk (memory level N+1)

page size = 5*8 = 40 bytes
what if we had an oracle (perfect index)?
page-based access & random access

query \( x < 7 \)

memory (memory level N)

disk (memory level N+1)

size = 120 bytes

page size = 5*8 = 40 bytes

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15
page-based access & random access

disk (memory level N+1)

memory (memory level N)

oracle

query $x < 7$

output

1, 5, 12, 24, 23

1, 5

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size $= 5 \times 8 = 40$ bytes

size $= 120$ bytes

$40$ bytes
Page-based access & random access

Memory (memory level N)
- Size = 120 bytes
- Query: \( x < 7 \)
- Oracle: 2, 7, 13, 9, 8
- Output: 1, 5

Disk (memory level N+1)
- Page size = 5*8 = 40 bytes
- 1, 5, 12, 24, 23
- 2, 7, 13, 9, 8
- 10, 11, 6, 14, 15
page-based access & random access

query $x < 7$

input

\[1, 5, 12, 24, 23\]

\[2, 7, 13, 9, 8\]

output

\[1, 5, 2\]

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 = \(5 \times 8 = 40\) bytes

$40$ bytes
page-based access & random access

`query x<7`

memory (memory level N)

| 1, 5, 12, 24, 23 | 2, 7, 13, 9, 8 |

output

1, 5, 2

size=120 bytes

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

memory (memory level N)

size=120 bytes

10, 11, 6, 14, 15

query $x < 7$

oracle

output

1, 5, 2

disk (memory level N+1)

size=80 bytes

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

memory (memory level N)

size = 120 bytes

query \( x < 7 \)

oracle

[10, 11, 6, 14, 15] [2, 7, 13, 9, 8] [1, 5, 2, 6]

output

disk (memory level N+1)

size = 120 bytes

page size = 5*8 = 40 bytes

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

oracle

[10, 11, 6, 14, 15] [2, 7, 13, 9, 8] [1, 5, 2, 6]
page-based access & random access

memory (memory level N)

size = 120 bytes

query \( x < 7 \)

oracle

output

\begin{align*}
10, 11, 6, 14, 15 & \quad 2, 7, 13, 9, 8 \quad 1, 5, 2, 6
\end{align*}

disk (memory level N+1)

\begin{align*}
1, 5, 12, 24, 23 & \quad 2, 7, 13, 9, 8 \quad 10, 11, 6, 14, 15
\end{align*}

page size = \( 5 \times 8 = 40 \) bytes

was the oracle helpful?

\( \text{oracle} \) was the oracle helpful?

\( \text{output} \)
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;

**random access**
read one block; consume it partially; discard it; (may re-use);
read random next;

hardware can predict and start prefetching
prefetching can exploit full memory/disk bandwidth

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)

query: value < x over an array of N slots
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

```
data
result
```
how to implement it?

```java
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?
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 99% qualifies?

how can we know?
branches (if statements) are bad for the processors, can we avoid them?

how to bring the values? (remember we have the positions)
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?
result = new array[data.size];
j=0;
for (i=0; i<data.size; i++)
  if (data[i]<x)
    result[j++] = i;

what about having multiple queries?

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

BOSTON UNIVERSITY
query: value < x
over an array of N slots

should I scan?

should I probe an index?

how to decide which one is best?

total data movement &
computation
how can I prepare?

1) Read background research material

2) Start going over the papers
what to do now?

A) read the syllabus and the website
B) register to piazza
C) register to gradescope
D) register for the presentation (early next week!)
E) start submitting paper reviews (week 3)
F) go over the project (next week will be available)
G) start working on the proposal (week 3)
survival guide

class website: [https://midas.bu.edu/classes/CS591A1/](https://midas.bu.edu/classes/CS591A1/)
piazza website: [http://piazza.com/bu/spring2020/cs591a1/](http://piazza.com/bu/spring2020/cs591a1/)
presentation registration: [https://tinyurl.com/S2020-CS591-presentations](https://tinyurl.com/S2020-CS591-presentations)
gradescope entry-code: 9568G3
office hours: Manos (Tu/Th, before class)
    Andy (M/W 3-4pm), Ju Hyoung (M 11am-noon / F 3-4pm)
material: papers available from BU network
CS 591: Data Systems Architectures

class 2

Data Systems 101

modern main-memory data systems

next week: &

semester project