CS 591: Data Systems Architectures

class 2

Data Systems 101

Prof. Manos Athanassoulis

http://manos.athanassoulis.net/classes/CS591
some reminders

- no smartphones
- no laptop
class effort summary

2 classes per week / OH 4 days per week

each student
1 presentation/discussion lead + 2 reviews per week
(5 long and the rest short, can skip 3)

systems or research project + mid-semester report

expect to work several hours every week
systems project
implementation-heavy C/C++ project
group of 1-2

research project
group of 3-4
pick a subject (list will be available)
design & analysis
experimentation
class timeline

Week 2 – register for presentations by 1/31  
first presentation on 2/5

Week 4 (or earlier) – form your groups by 2/15

Week 7 – submit your project report on 3/8

Week 15 – Project presentations submit all material by 4/30

discussions  
interaction in OH  
questions

discussions  
interaction in OH  
questions
Piazza

all discussions & announcements

http://piazza.com/bu/spring2019/cs591a1/
also available on class website

17 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

scientific experiments
social

sensors

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, what’s inside?
application/SQL access patterns complex queries

Indexing

Data

algorithms and operators

op

op

op

op

op
Memory Hierarchy

modules

- Query Parser
- Query Compiler
- Optimizer
- Evaluation Engine
- Memory/Storage Management
- Indexing
- Transaction Management

application/SQL access patterns complex queries

Memory Hierarchy
- CPU
- Caches
- Memory
- Disk
growing environment

db
- large systems
- complex
- lots of tuning
- legacy

noSQL
- simple, clean
- “just enough”

more complex applications

need for scalability

newSQL

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

[noSQL]
$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

- registers/CPU
  - my head
    - ~0
  - this room
    - 1min
- on chip cache
  - this building
    - 10min
- on board cache
  - Washington, DC
    - 5 hours
- memory
  - Pluto
    - 2 years
- disk
  - Andromeda
    - 2000 years
- tape
  - ~1min
  - this building
    - 10min
  - Washington, DC
    - 5 hours
  - Pluto
    - 2 years
  - Andromeda
    - 2000 years
memory hierarchy (by Jim Gray)

```
2x on chip cache
10x on board cache

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
- 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

Graph:
- Performance vs. Time
- CPU: Faster
- DRAM: Cheaper/Larger

Legend:
- CPU
- DRAM
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
- flash

Data go through all necessary levels

Also read unnecessary data

Need to read only X
Read the whole page

Photo from NBC
data movement

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

Data go through all necessary levels and also read unnecessary data.

Need to read only X page, not the whole 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)

size = 120 bytes

disk (memory level N+1)

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

page size = \( 5 \times 8 = 40 \) bytes
page-based access & random access

scan

query $x < 7$

memory (memory level N)

size = 120 bytes

1, 5, 12, 24, 23

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

$40$ bytes
page-based access & random access

<table>
<thead>
<tr>
<th>Memory Level N</th>
<th>Memory Level N+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>size=120 bytes</td>
<td>page size = 5*8 = 40 bytes</td>
</tr>
</tbody>
</table>

query $x < 7$

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

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

query $x < 7$

memory (memory level N)

size=120 bytes

output

1, 5, 2

disk (memory level N+1)

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

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

query $x < 7$

<table>
<thead>
<tr>
<th>Memory (memory level N)</th>
<th>Disk (memory level N+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>size = 120 bytes</td>
<td>page size = $5 \times 8 = 40$ bytes</td>
</tr>
<tr>
<td>1, 5, 12, 24, 23</td>
<td>10, 11, 6, 14, 15</td>
</tr>
<tr>
<td>2, 7, 13, 9, 8</td>
<td></td>
</tr>
</tbody>
</table>

output 1, 5, 2

$80$ bytes
page-based access & random access

query $x < 7$

scan

memory (memory level N)

size=120 bytes

10, 11, 6, 14, 15

disk (memory level N+1)

size=120 bytes

1, 5, 12, 24, 23

2, 7, 13, 9, 8

output

1, 5, 2

10, 11, 6, 14, 15

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

80 bytes
page-based access & random access

size=120 bytes
memory (memory level N)

disk (memory level N+1)

query \( x < 7 \)

scan

output

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

page size = 5*8 = 40 bytes

$80$ bytes
page-based access & random access

size=120 bytes
memory (memory level N)

disk (memory level N+1)

page size = 5*8 = 40 bytes

query \( x < 7 \)

scan

output

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

1, 5, 12, 24, 23
2, 7, 13, 9, 8
10, 11, 6, 14, 15
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 = $5 \times 8 = 40$ bytes
page-based access & random access

oracle
1, 5, 12, 24, 23

query \( x < 7 \)

output
1, 5

memory (memory level N)
size = 120 bytes

disk (memory level N+1)
page size = 5*8 = 40 bytes

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

memory (memory level N)

size=120 bytes

query \( x < 7 \)

oracle

output

| 1, 5, 12, 24, 23 | 2, 7, 13, 9, 8 | 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 | 40 bytes |
page-based access & random access

memory (memory level N)

size = 120 bytes

query $x < 7$

1, 5, 12, 24, 23

oracle

2, 7, 13, 9, 8

output

1, 5, 2

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

query $x < 7$
### Page-Based Access & Random Access

<table>
<thead>
<tr>
<th>Memory Level</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>1, 5, 12, 24, 23</td>
</tr>
<tr>
<td>N+1</td>
<td>2, 7, 13, 9, 8</td>
</tr>
<tr>
<td>Output</td>
<td>1, 5, 2</td>
</tr>
</tbody>
</table>

**Query x < 7**

- **Oracle**:
  - Memory (memory level N): 1, 5, 12, 24, 23
  - Disk (memory level N+1): 2, 7, 13, 9, 8

**Size = 120 bytes**

**Page Size = 5*8 = 40 bytes**

**$80$ bytes**
page-based access & random access

memory (memory level N)
- size = 120 bytes
- page size = 5*8 = 40 bytes

disk (memory level N+1)
- page size = 5*8 = 40 bytes

query \( x < 7 \)

oracle

output

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

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

memory (memory level N)
size=120 bytes

disk (memory level N+1)

query \( x < 7 \)

oracle

output

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

1, 5, 2, 6

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

input

$10, 11, 6, 14, 15$

$2, 7, 13, 9, 8$

output

$1, 5, 2, 6$

memory (memory level N)

size = 120 bytes

disk (memory level N+1)

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

$1, 5, 12, 24, 23$

$2, 7, 13, 9, 8$

$10, 11, 6, 14, 15$
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

BOSTON UNIVERSITY
how we store data
  layouts, indexes (classes 5,6)

every byte counts
  overheads and tradeoffs (classes 11,12)

know the query
  access path selection (class 10)
rules of thumb

**sequential access**
read one block; consume it completely; discard it; read next;

*hardware can predict and start prefetching*

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

*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

main-memory optimized-systems
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?

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

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;
```

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;

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

needs coordination!
what about result writing?
what about having multiple queries?

```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

Data

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
   - **Massively Parallel Databases and MapReduce Systems.** By Shivnath Babu and Herodotos Herodotou. Foundations and Trends in Databases, 2013

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 (now!)
E) start submitting paper reviews (week 3)
F) go over the project (end of this week will be available)
G) start working on the mid-semester report (week 3)
survival guide

class website: http://manos.athanassoulis.net/classes/CS591/
piazza website: http://piazza.com/bu/spring2019/cs591a1/
gradescope entry-code: MR7ZD4
office hours: Manos (Tu/Th, 2-3pm), Subhadeep (M/W 2-3pm)
material: papers available from BU network
class 2

Data Systems 101

modern main-memory data systems & semester project

next week: