data systems 101
prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS265/
quick recap
big data V’s
(it is not about size only)

volume  velocity  variety  veracity
declarative interface
ask “what” you want

the system decides “how” to best store and access data

data system
A data system **stores** data and **provides access** to data & **makes knowledge generation easy**
A data system stores data and provides access to data, making knowledge generation easy.
a data system **stores** data
and **provides access** to data

& makes knowledge generation easy
As apps become more complex and need to be more scalable, traditional databases (db) face issues with complex legacy tuning and being expensive. NoSQL databases offer a simpler, cleaner, and just enough solution.

NewSQL aims to combine the best of both worlds, offering a solution that is as flexible and powerful as NoSQL but with the reliability and ease of use of traditional databases.
2 classes per week - OH every day
1 presentation/discussion lead - 5 reviews
research (or systems) project
it all starts with how we store data

**every bit matters**
scale up vs scale out
performance - correctness - data models

using one machine as best as possible

using >1 machines as best as possible
logistics
how can I prepare?

1) start browsing some basic texts

**Get familiar with the very basics of traditional database architectures:**

**Get familiar with very basics of modern database architectures:**

**Get familiar with the very basics of modern large scale systems:**

2) play with basic data structures
implementation in C (linked list/hash table/tree)
my take on big data challenges

an example of design

hardware conscious designs

some basics on data systems

next time

modern main-memory optimized data systems
today
today

tomorrow
soon everyone will need to be a “data scientist”

hmm, my data is too big :(

how far away are we from a future where a data system sits in the critical path of everything we do?

new applications/requirements
data exploration

not always sure what we are looking for (until we find it)
daily data

years

[IBMbigdata]

data* skills

years

[StratosGuess]

data system design, set-up, tune, use
data systems that are **easy** to:

(years)

design & build
data systems that are **easy** to:

- design & build
- set-up & tune

(years)

(months)
data systems that are easy to:

- design & build (years)
- set-up & tune (months)
- use (hours/days)
data systems that are **easy** to:

- **design & build** (years)
- **set-up & tune** (months)
- **use** (hours/days)

**Examples:**
- adapt to new applications, new hardware, spin off alternative designs fast,
- create truly tailored solutions, enable more applications,
- bring development/set-up cost down (start-ups/scientific labs), etc…
my take on big data challenges

an example of design

hardware conscious designs

some basics on data systems
a “simple” example

assume an array of \( N \) integers:
find all positions where \( \text{value} > x \)

exists in all systems: sql, nosql, newsql

\( \text{select operator} \)
assume an array of $N$ integers:
find all positions where $value > x$

```
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```
assume an array of $N$ integers: find all positions where $value > x$

```java
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```

what if only 1% qualifies?
assume an array of $N$ integers: find all positions where $value > x$

res = new array[data.size]

$j = 0$
for ($i = 0; i < data.size; i++$)
    if $data[i] > x$
        $res[j++] = i$

what if only 1% qualifies?
assume an array of $N$ integers: find all positions where $value > x$

```java
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```

what if only 1% qualifies?
assume an array of $N$ integers: find all positions where $value > x$

```java
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```

what if only 1% qualifies?
assume an array of \( N \) integers: find all positions where \( value > x \)

```java
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```

what if only 1% qualifies?
assume an array of $N$ integers: find all positions where $value > x$

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

memory
assume an array of $N$ integers: find all positions where $value > x$

```java
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```

what if only 1% qualifies?
assume an array of $N$ integers: find all positions where $value > x$

```java
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```

what if only 1% qualifies?
assume an array of $N$ integers: find all positions where $\text{value} > x$

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

what if only 1% qualifies?
assume an array of \( N \) integers: find all positions where \( value > x \)

\[
\begin{align*}
\text{res} &= \text{new array}[\text{data.size}] \\
j &= 0 \\
\text{for} \ (i=0; \ i<\text{data.size}; \ i++) \\
&\quad \text{if data}[i] > x \\
&\quad \quad \text{res}[j++] = i
\end{align*}
\]

what if only 1% qualifies?
assume an array of \( N \) integers: find all positions where \( \text{value} > x \)

```
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
  if data[i] > x
    res[j++] = i
what if only 1% qualifies?
```

memory
Data memory

```
assume an array of \( N \) integers: find all positions where \( \text{value}>x \)
```

```
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```

```
what if only 1% qualifies?
```

```
but how can we know?
```

memory

```
data

copy res
```

CS265, Spring 2016
Stratos Idreos
assume an array of $N$ integers: find all positions where $value > x$

$$res = \text{new array}[\text{data.size}]$$

$j = 0$

for ($i = 0; i < \text{data.size}; i++$)
  if $\text{data}[i] > x$
    $res[j++] = i$

what if 90% qualifies?

result size = qualifying values $\times x$ bytes
assume an array of \( N \) integers: find all positions where \( \text{value} > x \)

\[
\text{res}=\text{new array}[\text{data.size}]
\]

\[
j=0
\]

\[
\text{for} \ (i=0; \ i<\text{data.size}; \ i++)
\]

\[
\quad \text{if} \ \text{data}[i]>x
\]

\[
\quad \text{res}[j++]=i
\]

what if 90% qualifies?

result size = qualifying values \( \times \) x bytes

bit vector for res?

\[
\begin{array}{ccccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1
\end{array}
\]

vs

\[
\begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1
\end{array}
\]
assume an array of $N$ integers:
find all positions where $value > x$

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

what if 90% qualifies?

result size = qualifying values $\times$ x bytes

bit vector for res?

1 0 0 0 1 1 0 1
0 0 0 0 1 0 1 0
0 0 0 0 1 1 0 0
0 0 0 0 1 1 1 1
0 0 0 1 0 0 1 0

if statements = bad, bad, bad
assume an array of \( N \) integers: find all positions where \( \text{value} > x \)

and we haven’t even started discussing about how to find the qualifying values…

what if 90% qualifies?

result size = qualifying values \( \times \) x bytes

bit vector for res?

if statements = bad, bad, bad

res = new array[data.size]

\[
\begin{align*}
    j &= 0 \\
    \text{for} \ (i=0; \ i<\text{data.size}; \ i++) \\
    & \quad \text{if} \ \text{data}[i] > x \\
    & \quad \text{res}[j++] = i
\end{align*}
\]
assume an array of $N$ integers: find all positions where $value > x$

```
res = new array[data.size]
j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```
assume an array of $N$ integers:
find all positions where $value > x$

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

NUMA architectures?
SIMD functionality?
& what about result writing?

not as simple as spinning off $N$ threads…
assume an array of $N$ integers:
find all positions where $value > x$

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

$N >> 1$ queries in parallel

q1, q2, q3
assume an array of \( N \) integers: find all positions where \( value > x \)

```java
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i

N >> 1 queries in parallel

q1, q2, q3
```
assume an array of $N$ integers: find all positions where value $> x$

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

$N >> 1$ queries in parallel
assume an array of $N$ integers: find all positions where $value>x$

```java
res = new array[data.size]

j = 0
for (i = 0; i < data.size; i++)
    if data[i] > x
        res[j++] = i
```

$N \gg 1$ queries in parallel
assume an array of $N$ integers:
find all positions where $value > x$

res = new array[data.size]

$j = 0$

for ($i = 0; i < data.size; i++$)
  if data[$i$] > $x$
    res[$j++$] = $i$

$q_1, q_2, q_3, q_4$

$N >> 1$ queries in parallel
assume an array of $N$ integers:
find all positions where $value > x$

\[
\text{res} = \text{new array}[10] \\
j = 0 \\
\text{for } (i = 0; i < 10; i++) \\
\quad \text{if } data[i] > x \text{ res}[j++] = i
\]
cost: data touched & computation

![Graph showing speed, CPU, and memory over time.](image)
assume an array of $N$ integers: find all positions where $\text{value} > x$

option 1: **scan** all data

option 2: use a **tree** (do not consider tree generation costs)

which one is best?
my take on big data challenges

an example of design

hardware conscious designs

some basics on data systems
system where db runs

- cpu - cpu - cpu - cpu
- cpu registers
- caches
- memory
- disk - disk - disk - disk

memory hierarchy

+ flash
+ non volatile memory
Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD Edgar F. Codd Innovations award

100Kx disk
Pluto 2 years

100x memory
New York 1.5 hours

10x on board cache
this building 10 min

2x on chip cache
this room 1 min

registers
my head ~0
- CPU
- registers
- on chip cache
- on board cache
- memory
- DRAM
- disk

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

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

- SRAM: ~1ns
- DRAM: ~100ns

The diagram illustrates the concept of the "memory wall" and the hierarchy of data storage devices from fastest to slowest: registers, on-chip cache, on-board cache, memory, and disk. The memory wall is highlighted by the slower access times of memory and disk compared to the faster access times of registers and cache.

- **faster**
- **cheaper**

The speed of access is represented by the graph, with faster devices on the left and cheaper devices on the bottom.
Design of storage/access methods/algorithms should minimize: data misses + instruction misses
random access & page-based access

data value $x$

need to only read $x$...
but have to read all of page 1

page1 page2 page3 ...
query $x < 5$

(size = 120 bytes)

memory level N

memory level N-1

\[
\begin{array}{cccccc}
5 & 10 & 6 & 4 & 12 \\
2 & 8 & 9 & 7 & 6 \\
7 & 11 & 3 & 9 & 6 \\
\end{array}
\]

page size: 5x8 bytes
query \( x < 5 \)

\[ \text{memory level N} \]

\[ \text{scan} \rightarrow \]

\[ 5 \ 10 \ 6 \ 4 \ 12 \]

\( \text{size}=120 \text{ bytes} \)

\[ \text{memory level N-1} \]

\[ 5 \ 10 \ 6 \ 4 \ 12 \ \ 2 \ 8 \ 9 \ 7 \ 6 \ \ 7 \ 11 \ 3 \ 9 \ 6 \ ... \]

page size: 5x8 bytes
query $x < 5$

memory level $N$

(scan)

(size=120 bytes)

memory level $N-1$

page size: 5x8 bytes
query \( x < 5 \)

```
5 10 6 4 12
2 8 9 7 6
7 11 3 9 6
```

memory level \( N \)

```
5 10 6 4 12
```

scan

(memory level \( N-1 \))

```
5 10 6 4 12
2 8 9 7 6
7 11 3 9 6
```

page size: 5x8 bytes

(size=120 bytes)

40 bytes
**query** $x<5$

Scan:

- Memory level $N$
  - 5 10 6 4 12
  - 2 8 9 7 6
  - 4

Size = 120 bytes

Memory level $N-1$

- 5 10 6 4 12
- 2 8 9 7 6
- 7 11 3 9 6
- ...

Page size: 5x8 bytes
query $x < 5$

memory level $N$

(size = 120 bytes)

memory level $N-1$

page size: 5x8 bytes
query $x < 5$

memory level $N$

(size=120 bytes)

memory level $N-1$

page size: 5x8 bytes
query $x < 5$

(memory level $N$)

[size=120 bytes]

(memory level $N-1$)

[page size: 5x8 bytes]
query $x < 5$

Scan:

memory level $N$

(size=120 bytes)

memory level $N-1$

Page size: 5x8 bytes

...
Scan

\(x < 5\)

memory level \(N\)

(size=120 bytes)

memory level \(N-1\)

page size: 5x8 bytes

80 bytes
query \( x < 5 \)

Scan:

Memory level N:
- 7 11 3 9 6
- 2 8 9 7 6
- 4 2 3

(size=120 bytes)

Memory level N-1:
- 5 10 6 4 12
- 2 8 9 7 6
- 7 11 3 9 6

PageSize: 5x8 bytes
an oracle gives us the positions

\[ \text{query } x < 5 \]

(memory level \(N\))

(page size: 5x8 bytes)

(memory level \(N-1\))

(size=120 bytes)
an oracle gives us the positions

\[ \text{query } x < 5 \]

<table>
<thead>
<tr>
<th>oracle</th>
<th>memory level N</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 10 6 4 12</td>
<td>(size=120 bytes)</td>
</tr>
</tbody>
</table>

memory level N-1

| 5 10 6 4 12 | 2 8 9 7 6 | 7 11 3 9 6 | ... |

page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

oracle

(size=120 bytes)
memory level N

5 10 6 4 12

4

memory level N-1

5 10 6 4 12

2 8 9 7 6

7 11 3 9 6

... page size: 5x8 bytes
an oracle gives us the positions

\[ \text{query } x < 5 \]

(size=120 bytes)

memory level N

\[ \begin{array}{cccccc}
5 & 10 & 6 & 4 & 12 & 4
\end{array} \]

memory level N-1

\[ \begin{array}{cccccc}
5 & 10 & 6 & 4 & 12 & 2 & 8 & 9 & 7 & 6 & 7 & 11 & 3 & 9 & 6 & \ldots
\end{array} \]

page size: 5x8 bytes
an oracle gives us the positions

\textbf{query} \( x < 5 \)

\begin{align*}
\text{memory level N} & \quad \text{oracle} \\
\{5, 10, 6, 4, 12\} & \quad \{2, 8, 9, 7, 6\} \\
\text{(size=120 bytes)} & \quad 4
\end{align*}

\begin{align*}
\text{memory level N-1} & \\
\{5, 10, 6, 4, 12\} & \quad \{2, 8, 9, 7, 6\} & \quad \{7, 11, 3, 9, 6\} \\
\text{page size: 5x8 bytes}
\end{align*}
an oracle gives us the positions

query $x < 5$

(memory level N)

(size=120 bytes)

memory level N-1

page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

(memory level N)

(size=120 bytes)

oracle

5 10 6 4 12

oracle

2 8 9 7 6

4 2

(memory level N-1)

5 10 6 4 12

2 8 9 7 6

7 11 3 9 6

page size: 5x8 bytes

80 bytes
an oracle gives us the positions

query $x < 5$

(memory level $N$

(size=120 bytes)

memory level $N-1$

page size: 5x8 bytes

80 bytes
an oracle gives us the positions

query $x < 5$

 oracle

(size=120 bytes)
memory level $N$

memory level $N-1$

page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

(memory level $N$)

(size=120 bytes)

(memory level $N-1$)

page size: 5x8 bytes
an oracle gives us the positions

query \( x < 5 \)

(size=120 bytes)
memory level \( N \)

oracle

\[
\begin{array}{cccccc}
7 & 1 & 1 & 3 & 9 & 6 \\
2 & 8 & 9 & 7 & 6 & 4 & 2 & 3
\end{array}
\]

memory level \( N-1 \)

\[
\begin{array}{cccccc}
5 & 1 & 0 & 6 & 4 & 12 \\
2 & 8 & 9 & 7 & 6 & 7 & 1 & 1 & 3 & 9 & 6 \\
\ldots
\end{array}
\]

page size: 5x8 bytes
when does it make sense to have an oracle
how can we minimize the cost

e.g., \textbf{query} \texttt{x<5}

\begin{tabular}{ccc}
5 & 10 & 6 \\
4 & 12 & 2 \\
8 & 9 & 7 \\
9 & 6 & 7 \\
7 & 11 & 3 \\
9 & 6 & 1
\end{tabular}
sequential access:
read one block; consume it completely; discard it; read next

in parallel/prefetching

what is next?

hardware can better predict-buffer sequential pages to be read
e.g., 2MB buffers in modern DRAM
amortize cost of moving disk arms
random access:
read one block; consume it partially; discard it; might have to read it again in future; read “random” next;
C/C++
MAIN-MEMORY OPTIMIZED DATA SYSTEMS
MAIN-MEMORY OPTIMIZED DATA SYSTEMS
MAIN-MEMORY OPTIMIZED DATA SYSTEMS
my take on big data challenges

an example of design

hardware conscious designs

some basics on data systems
design

- **logical** design
- **physical** design
- **system** design
essential steps in using a database system

1. **clean**
2. **schema**
3. **load**
4. **tune**

---

Experts/system admins

---

User/apps

---

query
relational model + SQL

- **professors**
  - (id, name, …)
  - key
  - table/relation

- **courses**
  - (id, name, profId, …)
  - column/attribute

- **students**
  - (id, name, …)
  - table/relation

- **database**
relational model + SQL

create table for professors:

CREATE TABLE professors (id:integer, name: char(40), telephone: char(10), ...)
create table for professors:

**create table** professors (id:integer, name: char(40), telephone: char(10), …)

**insert into** professors (76897689, “john smith”, …)
create table for professors:

**create table** professors (id:integer, name: char(40), telephone: char(10), …)

**insert into** professors (76897689, “john smith”, …)

give me the names of all students:

**select** name **from** students
relational model + SQL

create table for professors:

\[
\text{create table professors (id:integer, name: char(40), telephone: char(10), \ldots)}
\]

insert into professors (76897689, “john smith”, \ldots)

give me the names of all students:

\[
\text{select name from students where GPA>3.0}
\]
**Schema**

employee
(id:int, name:varchar(50), office:char(5), telephone:char(10), city:varchar(30), salary:int)

**Data**

(1, name1, office1, tel1, city1, salary1)
(2, name2, office2, tel2, city2, salary2)
(3, name3, office3, tel3, city3, salary3)
(4, name4, office4, tel4, city4, salary4)
(5, name5, office5, tel5, city5, salary5)
(6, name6, office6, tel6, city6, salary6)
(7, name7, office7, tel7, city7, salary7)
(8, name8, office8, tel8, city8, salary8)
(9, name9, office9, NULL, city9, salary9)

SQL: insert into employee
(1, name1, office1, tel1, city1, salary1)

Cardinality = 9

Value does not exist
give me all students enrolled in cs265
relational model + SQL

give me all students enrolled in cs265
give me all students enrolled in cs265
give me all students enrolled in cs265

```sql
select student.name from students, enrolled, courses where
courses.name="cs165" and enrolled.courseld=course.id and
student.id=enrolled.studentId
```
relational model + SQL

The database consists of three tables:

- **professors**: (id, name, ...)
- **courses**: (id, name, profId, ...)
- **students**: (id, name, ...)

The `enrolled` table is a join table and is represented as:

**enrolled**: (studentId, courseId, ...)

**foreign key**

**give me all students enrolled in cs265**

```sql
select student.name
from students, enrolled, courses
where courses.name = "cs165" and enrolled.courseld = course.id and student.id = enrolled.studentId
```
star schema

fact table
(id1, id2, …)

dimension table 1
(id1, …)

dimension table 2
(id2, …)

…

…

…
essential steps in using a database system

- clean
- schema
- load
- query
- tune

Experts/system admins

User/apps
declarative interface
ask what you want

so do db systems “just work”?
declarative interface
ask what you want

indexes/views/tuning knobs

db system
declarative interface
ask what you want

indexes/views/tuning knobs

but … db cracking, adaptive* ideas

db system
design

logical design

physical design

system design
\textbf{select} \texttt{min(A) from R where B<10 and C<80}
two more lectures next week and then we go into discussion mode

wed: db architectures basics
fri: projects

next week: paper signup + systems project will be online