data systems 101
prof. Stratos Idreos
HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS265/
big data V’s
(it is not about size only)

volume    velocity    variety    veracity

actually none of that is really new...

ew:
our ability to gather and store machine generated data
broad understanding that we cannot just manually get value out of data
a data system **stores** data
and **provides access** to data

& makes knowledge generation easy
Database kernel

Data

Algorithms/operators

Applications

SQL

cpu
memory
disk

CS265, Spring 2017
Stratos Idreos

4 /55
declarative interface
ask “what” you want

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

data system
SQL
complex legacy tuning expensive ...

noSQL
simple clean just enough ...

as apps become more complex
as apps need to be more scalable

newSQL

Gartner: DBMS Market = ~36Billion
noSQL/Hadoop = ~700Million
SQL = ~rest

goal: understand why, how and what is next

more data

more applications

the need for new systems

more h/w
soon everyone will need to be a “data scientist”

new applications/requirements

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?
data exploration

not always sure what we are looking for (until we find it)
system where db runs

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

+ flash
+ non volatile memory

memory hierarchy
Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD Edgar F. Codd Innovations award
The term static differentiates SRAM from DRAM (dynamic random-access memory) which must be periodically refreshed. SRAM is faster and more expensive than DRAM; it is typically used for CPU cache while DRAM is used for a computer's main memory.
design of storage/access methods/algorithms should minimize:  

\[ \text{data misses} + \text{instruction misses} \]
random access & page-based access

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

data value $x$

page1  page2  page3  …
query $x < 5$

memory level $N$

scan

scan

(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

try to read something and use it fully and never ever read it again to avoid data transfer costs - reading data from disk has been the major effort
An oracle gives us the positions

$$\text{query } x < 5$$

<table>
<thead>
<tr>
<th>oracle</th>
<th>oracle</th>
<th>4 2 3</th>
</tr>
</thead>
</table>

(size=120 bytes)

Memory level N

<table>
<thead>
<tr>
<th>5 10 6 4 12</th>
<th>2 8 9 7 6</th>
<th>7 11 3 9 6</th>
</tr>
</thead>
</table>

Page size: 5x8 bytes

Memory level N-1

it does not make a difference - in fact we have to query and maintain the oracle
when does it make sense to have an oracle
how can we minimize the cost

e.g., query $x < 5$

5 10 6 4 12  2 8 9 7 6  7 11 3 9 6  ...
it all starts with how we store data

**every bit matters**
**sequential access:**
read one block; consume it completely; discard it; read **next**

in parallel/prefetching

what is next?

1 2 3 4

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;

disk mechanical arms - high cost to move arms - so when you move them exploit it and read everything in there memory buffers
C/C++
a “simple” example

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

qualifying positions

select operator

exists in all systems: sql, nosql, newsql

make it like a key-value store
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?

but how can we know?

memory

data

copy res

assume an array of $N$ integers: find all positions where \texttt{value} > x

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

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

\textbf{what if 90\% qualifies?}

result size = qualifying values * x bytes

bit vector for res?

\begin{verbatim}
00000101
00000110
00000111
00001001
\end{verbatim}

\textbf{vs}

\begin{verbatim}
1000111101
\end{verbatim}

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

```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 $value > x$

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

```
res = new array[10]
j = 0
for (i = 0; i < 10; i++)
  if data[i] > x
    res[j++] = i
```
assume an array of $N$ integers: find all positions where $value > x$

- option 1: **scan** all data
- option 2: use a **tree** (do not consider tree generation costs)

which one is best
cost: data touched & computation
a "simple" example

**build a key-value store**
similar to the ones Facebook, Google, etc use

interface supported: put, get, scan, count, get range, load
unique key-value pairs, $r>>w$ but $w>>0$

data

how to store and access
design

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

1. clean
2. schema
3. load
4. tune
5. query

experts/system admins

user/apps
relational model + SQL

create table for professors:
\textbf{create table} professors (id:integer, name: char(40), telephone: char(10), …)

\textbf{insert into} professors (76897689, “john smith”, …)

give me the names of all students:
\textbf{select} name \textbf{from} students \textbf{where} GPA\textgreater{}3.0
**employee**
(id: int, name: varchar(50), office: char(5), telephone: char(10), city: varchar(30), salary: int)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>name1</td>
<td>office1</td>
<td>tel1</td>
<td>city1</td>
<td>salary1</td>
</tr>
<tr>
<td>2</td>
<td>name2</td>
<td>office2</td>
<td>tel2</td>
<td>city2</td>
<td>salary2</td>
</tr>
<tr>
<td>3</td>
<td>name3</td>
<td>office3</td>
<td>tel3</td>
<td>city3</td>
<td>salary3</td>
</tr>
<tr>
<td>4</td>
<td>name4</td>
<td>office4</td>
<td>tel4</td>
<td>city4</td>
<td>salary4</td>
</tr>
<tr>
<td>5</td>
<td>name5</td>
<td>office5</td>
<td>tel5</td>
<td>city5</td>
<td>salary5</td>
</tr>
<tr>
<td>6</td>
<td>name6</td>
<td>office6</td>
<td>tel6</td>
<td>city6</td>
<td>salary6</td>
</tr>
<tr>
<td>7</td>
<td>name7</td>
<td>office7</td>
<td>tel7</td>
<td>city7</td>
<td>salary7</td>
</tr>
<tr>
<td>8</td>
<td>name8</td>
<td>office8</td>
<td>tel8</td>
<td>city8</td>
<td>salary8</td>
</tr>
<tr>
<td>9</td>
<td>name9</td>
<td>office9</td>
<td>NULL</td>
<td>city9</td>
<td>salary9</td>
</tr>
</tbody>
</table>

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

Cardinality = 9

Value does not exist
give me all students enrolled in cs265

```sql
select student.name from students, enrolled, courses
where courses.name = 'cs165' and enrolled.courseId = course.id
and student.id = enrolled.studentId
```
A star schema consists of a fact table and multiple dimension tables. The fact table contains fact data, while each dimension table contains dimension data. The fact table has a many-to-many relationship with the dimension tables.

- Fact table: (id1, id2, ...)
- Dimension table 1: (id1, ...)
- Dimension table 2: (id2, ...)
key-value store vs relational

can we store a document collection in a relational systems?
can we store a relational database in a key-value store?
design

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

1. clean
2. schema
3. load
4. tune
5. query

Experts/system admins

User/apps
declarative interface
ask what you want

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

DBA
indexes/views/tuning knobs

but ... db cracking, adaptive* ideas
design

- logical design
- physical design
- system design
select min(A) from R where B<10 and C<80
logistics
projects

**option 1: systems project**
basic key-value store functionality - work individually
single machine - multi-core design
basic design as in Facebook, LinkedIn, Mongo, etc.
can lead to research

**option 2: research project**
self-designing data systems + shape-shifting access methods
research with DASlab researchers - groups of 3
available for cs165 students or otherwise advanced students
next generation adaptive Key-value stores (with Facebook)
C/C++

no libraries unless we explicitly allow it
we expect you build everything from scratch
so you can control storage and access 100%
midway check-in (10%)

special class (2-3 hour?) in mid March:
1) design docs
2) at least one performance example
3) presentation/poster
two reading sessions and two hacking sessions per week? so maybe a minimum of 15 hours per week (if you already have decent hacking and data structure/algorithms experience)
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)
two more lectures next week and then we go into discussion mode

wed: db architectures basics
fri: projects

next week: paper signup + systems project is already online
Action steps:
1) Read the syllabus & website carefully,
2) Register to Piazza,
3) Do P0 if you have not taken CS165 and check self-test,
4) Register for paper presentation (week 2),
5) Start submitting your paper reviews (week 3)

web site: http://daslab.seas.harvard.edu/classes/cs265/
piazza: piazza.com/harvard/spring2017/cs265/home
office hours: Stratos: Wed/Thur/Fri, 3-4pm, MD139
TF office hours: Mon ?, Tue, 3-4pm, MD 136
textbook: nope
research papers will be available from the Harvard network
data systems 101
BIG DATA SYSTEMS
prof. Stratos Idreos

next time
modern main-memory
optimized data systems