summary & data analytics without loading
prof. Stratos Idreos
HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS265/
today

summary of concepts so far

how to make the most of the class

next steps

no more data loading
MAIN-MEMORY & MASSIVELY SCALABLE DATA SYSTEMS
scale up vs scale out

performance - correctness - data models

using one machine as best as possible

using >1 machines as best as possible
using one machine as best as possible

scale up vs scale out

performance - correctness - data models

but we have been discussing a limited set of common concepts

using >1 machines as best as possible
declarative interface
ask "what" you want

the system decides
"how" to best store
and access data

data system
logical design

physical design

system design
data systems architectures

data structures + algorithms

data

some problems:
how to store data
how to access data

how to best answer a complex query (e.g., which data to access first and how)

how to answer millions of queries concurrently

how to guarantee correctness and availability

how to spend the least possible energy

...
it all starts with how we store data

**every bit matters**
it all starts with how we store data

**every bit matters**

e.g., morsels and google papers

partitioning

NUMA in one machine &

distributed processing
single machine

shared nothing vs shared everything
read vs write tradeoffs
single machine

shared nothing vs shared everything
read vs write tradeoffs
partition data across **sockets** to minimize communication

**single machine**

shared nothing vs shared everything
read vs write tradeoffs
>1 machines

shared nothing vs shared everything
read vs write tradeoffs
>1 machines

shared nothing vs shared everything
read vs write tradeoffs
Partition data across **nodes** to minimize communication.

>1 machines

Shared nothing vs shared everything

Read vs write tradeoffs
>1 clusters

shared nothing vs shared everything
read vs write tradeoffs
>1 clusters

shared nothing vs shared everything
read vs write tradeoffs
partition data across clusters to minimize communication

>1 clusters

shared nothing vs shared everything read vs write tradeoffs
As apps become more complex, as apps need to be more scalable, pushing existing concepts into large scale systems by sacrificing functionality/properties (at first)
as apps become more **complex**
as apps need to be more **scalable**

pushing existing concepts into large scale systems by sacrificing functionality/properties (at first)

it all starts with how we store data
As apps become more complex, as apps need to be more scalable, it all starts with how we store data. **NewSQL** is pushing existing concepts into large scale systems by sacrificing functionality/properties (at first).
hardware is not just hardware

computation/algorithms/data structures

data management

it all starts with how we store data

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

data has always been big

volume  velocity  variety  veracity
data systems today
allow us to answer queries fast

data systems tomorrow
should allow us to find fast which queries to ask
1. INTRODUCTION

Adaptive storage; adaptive hybrids; dynamic operators

Keywords

Algorithms, Design, Performance

General Terms

H.2.2, H.2.4

Categories and Subject Descriptors

Database Management

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or re-arrange, or to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or re-arrange, or to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or re-arrange, or to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or re-arrange, or to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or re-arrange, or to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
too many combinations to maintain in parallel
\[
q(L) = \sum_{i=1}^{\left|L\right|} \max(cost_{i}^{IO}, cost_{i}^{CPU})
\]

for a given query we can know exactly which layout is best

the one that will cause the fewer cache misses
if we know all queries upfront we can choose the layouts

adaptive storage:
continuously adapt layouts based on incoming queries
but computing all possible combinations is expensive…

query

```
select A+B+C+D from R where A<10 and E>10
```

1. deal only with attributes referenced in queries
2. handle select clause separately from where clause
3. start from pure column-store and build up
4. stop when no improvement possible
initialization

querying

storage
initialization

querying

storage

Continuous Improvement
every query is treated as an advice on how data should be stored
Every query is treated as an advice on how data should be stored.
same concept applies in drastically different models
e.g., data series
time-series indexing

\( \text{time-series} = \{a_1, a_2, \ldots, a_n\} \)

typical query: find a time-series which is similar to time series \( x \)
time-series indexing

\[ \text{time-series} = \{a_1, a_2, \ldots, a_n\} \]

typical query: find a time-series which is similar to time series \( x \)

![Diagram illustrating time-series indexing](image)
time-series indexing

time-series={a1,a2,...,an}
typical query: find a time-series which is similar to time series x
time-series indexing

time-series={a1,a2,...,an}

typical query: find a time-series which is similar to time series x

we build the index (layout) only when we get queries
it all starts with how we store data
e.g., adaptive denormalization

star schema

fact table
(id1, id2, ...)

dimension table 1
(id1, ...)

dimension table 2
(id2, ...)

...
e.g., adaptive denormalization

- Populate based on incoming query patterns

- Denormalized fragments: queries only need to fast scan

normalized data

possible denormalized space
how to make the most of it
no such thing as wrong questions/answers

the more you ask/participate the more you learn
the more you ask/participate the more you learn

no such thing as wrong questions/answers

MORE BRAINSTORMING STYLE DISCUSSIONS
read both P and B papers before class
READING   SLIDES   REVIEWS

what is the problem
why is it important
why is it hard
why existing solutions do not work
what is the core intuition for the solution
solution step by step
does the paper prove its claims
exact setup of analysis/experiments
are there any gaps in the logic/proof
possible next steps
presentations

answer review questions

no text unless really needed

use graphics to express concepts by example
OFFICE HOURS daily, drop by!
### coming up

1. more research lectures on bitmap indexing and interactive analytics

2. class papers on big machine learning, newSQL (spark, flink), polysystems, data structures

3. project evaluations, posters
ADAPTIVE LOADING

Here are my Data Files. Here are my Queries. Where are my Results?
S. Idreos, I. Alagiannis, R. Johnson, and A. Ailamaki.
In Proceedings of the 5th International Conference on Innovative Data Systems Research (CIDR), 2011

NoDB: efficient query execution on raw data files
I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki

NoDB: efficient query execution on raw data files
I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki
Communications of the ACM, Research Highlights, 2015
too many preparation options lead to complex installation

expert users - idle time - workload knowledge != exploration, easy
users/applications
declarative interface
ask what you want

DBA

db system
users/applications

need to choose the proper system & workloads/applications change rapidly

DBA1

DBA2

data system 1

data system 2

...
be able to query the data immediately & with good performance
be able to query the data immediately & with good performance

explore data and gain knowledge “immediately”
initialization

querying

loading
every query is treated as an advice on how data should be stored
continuous, lightweight actions to co-locate relevant data

every query is treated as an advice on how data should be stored
copy data inside the database

database now has full control

load
tune
query

load

slow process...
not all data might be needed all the time
1 file, 4 attributes, 1 billion tuples

**Database vs. Unix tools**

- **DB**
- **Awk**

<table>
<thead>
<tr>
<th>single query cost (secs)</th>
<th>0</th>
<th>550</th>
<th>1,100</th>
<th>1,650</th>
<th>2,200</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Awk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 file, 4 attributes, 1 billion tuples

break down db cost

- Loading: 7%
- Query Processing: 93%

single query cost (secs)

- DB: 0-1,650 seconds
- Awk: 1,650-2,200 seconds
database vs. unix tools

1 file, 4 attributes, 1 billion tuples

break down db cost
- Loading 7%
- Query Processing 93%

loading is a major bottleneck
database vs. unix tools

1 file, 4 attributes, 1 billion tuples

break down db cost
- Loading: 7%
- Query Processing: 93%

loading is a major bottleneck

but writing/maintaining scripts does not scale
adaptive loading

load/touch only what is needed and only when it is needed
but raw data access is expensive
tokenizing - parsing - no indexing - no statistics

challenge: fast raw data access
query plan
query plan

scan
query plan

scan

db
query plan

scan

files

access raw data adaptively on-the-fly

loading
query plan

scan

files

cache

access raw data adaptively on-the-fly

loading
selective parsing
file indexing
file splitting
online statistics

query plan

scan

files

access raw data
adaptively on-the-fly

loading

cache
File → Partition → Break lines → Tokenize → Fill columns
selective tokenizing and parsing
indexing on raw files
known position

file
file

known position

looking for
known position looking for

file

point
file

known position

looking for
Single file

File1 File2 File3 FileN

split while parsing
The graph compares the execution times of MySQL, DBMS X, PostgreSQL, and PostgresRaw PM + C. The execution times are depicted for different queries (Q1, Q2, Q3, ..., Q20). MySQL shows significantly higher execution times compared to the other systems.
reducing data-to-query time
summary &
data analytics without loading

BIG DATA SYSTEMS

prof. Stratos Idreos