class 22

auto-tuning database kernels 2.0

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
a bit more about auto-tuning db kernels
and then a couple of open research topics
be able to query the data immediately & with good performance

explore data and gain knowledge “immediately”
database cracking
database cracking

- idle time
- workload knowledge
- external tools
- human control
database cracking
auto-tuning database kernels
incremental, adaptive, partial indexing

idle time
workload knowledge
external tools
human control
database cracking
auto-tuning database kernels
incremental, adaptive, partial indexing

initialization
querying
indexing

idle time
workload knowledge
external tools
human control
database cracking
auto-tuning database kernels
incremental, adaptive, partial indexing

idle time
workload knowledge
external tools
human control

Continuous Improvement

Stratos Idreos
database cracking
auto-tuning database kernels
incremental, adaptive, partial indexing

every query is treated as an advice on how data should be stored
Q1:
select R.A
from R
where R.A>10
and R.A<14
Q1:
select R.A
from R
where R.A > 10
and R.A < 14
Q1: select R.A from R where R.A > 10 and R.A < 14
Q1: select R.A from R where R.A>10 and R.A<14
Q1: select R.A from R where R.A > 10 and R.A < 14
Q1: select R.A from R where R.A > 10 and R.A < 14

column A

| 13 | 16 | 4 | 9 | 2 | 12 | 7 | 1 | 19 | 3 | 14 | 11 | 16 | 19 | 14 |

result

piece1: A <= 10

piece2: 10 < A < 14

piece3: A >= 14

gain knowledge on how data is organized
gain knowledge on how data is organized

column A

Q1: select R.A from R
where R.A > 10 and R.A < 14

result

dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007
Q1: select R.A from R where R.A>10 and R.A<14

Q2: select R.A from R where R.A>7 and R.A<=16

dynamically/on-the-fly within the select-operator
Q1:
select R.A 
from R 
where R.A>10 
and R.A<14

Q2:
select R.A 
from R 
where R.A>7 
and R.A<=16

piece1: A<=10
piece2: 10<A<14
piece3: A>=14

dynamically/on-the-fly within the select-operator

Stratos Idreos
Database Cracking CIDR 2007
Q1:
select R.A from R
where R.A > 10
    and R.A < 14

Q2:
select R.A from R
where R.A > 7
    and R.A <= 16

dynamically/on-the-fly within the select-operator
Q1:
select R.A
from R
where R.A > 10
and R.A < 14

Q2:
select R.A
from R
where R.A > 7
and R.A <= 16

dynamically/on-the-fly within the select-operator
Q1: select R.A from R where R.A > 10 and R.A < 14

Q2: select R.A from R where R.A > 7 and R.A <= 16

dynamically/on-the-fly within the select-operator
Q1: select R.A from R where R.A > 10 and R.A < 14

Q2: select R.A from R where R.A > 7 and R.A <= 16

dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007
Q1:
select R.A
from R
where R.A > 10
and R.A < 14

Q2:
select R.A
from R
where R.A > 7
and R.A <= 16

dynamically/on-the-fly within the select-operator

the more we crack, the more we learn

Database Cracking CIDR 2007
set-up

100K random selections
random selectivity
random value ranges
in a 10 million integer column

Response time (secs)

Query sequence (x1000)

Scan

Crack

Full Index
set-up
100K random selections
random selectivity
random value ranges
in a 10 million integer column

almost no
initialization overhead

continuous adaptation

Response time (sec)

Query sequence (x1000)

Scan
Crack
Full Index

0.001
0.01
0.1
1
10
100
1000
10000
100000
set-up
100K random selections
random selectivity
random value ranges
in a 10 million integer column

almost no
initialization overhead

continuous improvement
set-up

100K random selections
random selectivity
random value ranges
in a 10 million integer column

almost no
initialization overhead

continuous improvement

Response time (secs)

Query sequence (x1000)
cracking databases

basics (CIDR07)

updates (SIGMOD07)

>1 columns (SIGMOD09)

storage-restrictions (SIGMOD09)

robustness (PVLDB12)

algorithms (PVLDB11)

benchmarking (TPCTC10)

concurrency control (PVLDB12)

time-series (SIGMOD14)

multi-cores (SIGMOD15)

efficiency (SIGMOD16)

adaptive storage (SIGMOD14)

hadoop (Yale/Saarland)

b-trees (HP Labs)
concurrency control

problem: read queries become write queries! (?)

goal: be able to crack for multiple queries in parallel
traditional indexing

adaptive indexing
write queries

traditional indexing

read queries

adaptive indexing

Concurrency Control, PVLDB 12
Change index contents and structure
write queries

only index structure changes
read queries

traditional indexing

adaptive indexing

Concurrency Control, PVLDB 12
no need for traditional **locks** = too heavy

short term **latches** = fast and release quickly

<table>
<thead>
<tr>
<th>Change index contents and structure</th>
<th>Only index structure changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write queries</td>
<td>Read queries</td>
</tr>
</tbody>
</table>

traditional indexing

adaptive indexing
traditional indexing

all or nothing
change index contents and structure
write queries

adaptive indexing
incremental and optional
only index structure changes
read queries

Concurrency Control, PVLDB 12
Stratos Idreos

traditional indexing

all or nothing

change index contents and structure

write queries

traditional indexing

incremental and optional

only index structure changes

read queries

adaptive indexing

Concurrency Control, PVLDB 12
traditional indexing

all or nothing

incremental and optional

change index contents and structure

only index structure changes

write queries

read queries

adaptive indexing

Concurrency Control, PVLDB 12
Traditional indexing vs. adaptive indexing:

- **All or nothing**
  - Change index contents and structure
  - Write queries
  - Traditional indexing

- **Incremental and optional**
  - Only index structure changes
  - Read queries
  - Adaptive indexing

Concurrency Control, PVLDB 12
Stratos Idreos

all or nothing

incremental and optional

change index contents and structure

only index structure changes

write queries

read queries

traditional indexing

adaptive indexing

Concurrency Control, PVLDB 12
all or nothing
change index contents and structure
write queries

traditional indexing

incremental and optional
only index structure changes
read queries

adaptive indexing
Stratos Idreos

traditional indexing

all or nothing

change index contents and structure

write queries

Concurrency Control, PVLDB 12

incremental and optional

only index structure changes

read queries

adaptive indexing
traditional indexing

impact stable storage

all or nothing

change index contents and structure

write queries

adaptive indexing

stable storage optional

incremental and optional

only index structure changes

read queries

Concurrency Control, PVLDB 12
Stratos Idreos

---

Impact stable storage

All or nothing

Change index contents and structure

Write queries

Traditional indexing

---

Stable storage optional

Incremental and optional

Only index structure changes

Read queries

Adaptive indexing

Concurrency Control, PVLDB 12
**traditional indexing**

- need to serialize
- impact stable storage
- all or nothing
- change index contents and structure
- write queries

**adaptive indexing**

- can execute in any order
- stable storage optional
- incremental and optional
- only index structure changes
- read queries

Concurrency Control, PVLDB 12

Stratos Idreos
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20

disk

memory

A<10 IDs

B

B<20 IDs

C minC

column lock and release as soon as an operator completes
need to latch only to be cracked pieces (max 2 per select)

select \([a,b]\)
piece locking

avl-tree

concurrent control

crack column
piece locking

avl-tree  crack column  crack select
Concurrency Control, PVLDB 12
Stratos Idreos

Concurrency Control, PVLDB 12
piece locking

avl-tree  crack column
piece locking

avl-tree crack column max

Concurrency Control, PVLDB 12
pieve locking

avl-tree \rightarrow crack column

rlock \rightarrow max
piece locking

avl-tree

crack column

max

rlock
Concurrent Control, PVLDB 12
Stratos Idreos

avl-tree | crack column

10
90
140
200
300

piece locking
Stratos Idreos

piece locking

avl-tree

10
90
140
200
300

crack column

230
65

crack select
piece locking

avl-tree

crack column

wlock

wlock

65

230

crack select
piece locking

avl-tree

crack column

wlock

r/wlock

wlock

10

90

140

200

300

65

230
Stratos Idreos

piecing locking

avl-tree

crack column

crack select

q1, q2, q3...qn

wlock

10

90

140

200

300

65

230

90

10
piece locking

avl-tree

crack column

wlock

65

q1, q2, q3... qn

230

wlock

10

90

140

200

300

20

30

70

80

90
piece locking

avl-tree  crack column  crack select

wlock       65       q1,q2,q3...qn

wlock       230

10  90  140  200  300

20  30  70  80

wlock

piece locking

Stratos Idreos
Rows: 100M
Query: select sum(A) from R where v1 < A1 < v2
Selectivity: 0.01%, Random, # of queries: 1024
Clients: 1-32, Machine: 4 cores

Adaptive indexing maintains its performance advantage.
Query: select sum(A) from R where v1 < A1 < v2
Selectivity: 0.01%, Random, # of queries: 1024
Clients: 8, Machine: 4 cores

adaptive indexing maintains the adaptive behavior
stochastic cracking
robustness
Stratos Idreos

adaptive indexing
Response time: X
Response time: X
Stratos Idreos

Response time: $X$

adaptive indexing

Response time: $1000X$

v2 v1 v5 v4 v3
Stochastic Cracking, PVLDB 12
Stochastic Cracking, PVLDB 12
select [15,55]
select [15,55]
select [15,55]
select [15,55]

10  20  30  40  50  60

select [15,55]
select [15,55]

10 20 30 40 50 60

select [15,55]
Stochastic Cracking, PVLDB 12
the amount of work for each query depends on the index state

the state of the index depends on past queries patterns

select [15,55]
column with 100 unique integers \([1,100]\)

good sequence
column with 100 unique integers \([1,100]\)

<50

q1

good sequence
column with 100 unique integers [1,100]

<50
q1

N

good sequence
column with 100 unique integers [1,100]

<50
q1

good sequence
column with 100 unique integers [1,100]

<25  q2  <50  q1

good sequence
column with 100 unique integers [1,100]

<25  <50
q2   q1

N
N/2

good sequence
column with 100 unique integers $[1, 100]$

<25  q2  <50  q1

good sequence

$N$
$N/2$
column with 100 unique integers [1,100]

<25  <50  <75
q2   q1   q3

good sequence

N
N/2
column with 100 unique integers \([1,100]\)

\[
\begin{align*}
&\text{<25} & \text{<50} & \text{<75} \\
&q_2 & q_1 & q_3
\end{align*}
\]

good sequence

N
N/2
N/2
column with 100 unique integers [1,100]

<25 q2 <50 q1 <75 q3

good sequence

N
N/2
N/2

Stochastic Cracking, PVLDB 12
column with 100 unique integers [1,100]

<25  <50  <75
q2   q1   q3

good sequence

N  N/2  N/2

Stochastic Cracking, PVLDB 12
column with 100 unique integers [1,100]

<25  <50  <75
q2   q1   q3

good sequence

bad sequence

N
N/2
N/2
column with 100 unique integers [1,100]

- good sequence
  - \(<25\)
  - \(\text{q2}\)
  - \(<50\)
  - \(\text{q1}\)
  - \(<75\)
  - \(\text{q3}\)

- bad sequence
  - \(<2\)
  - \(\text{q1}\)
column with 100 unique integers [1,100]

<25  q2  <50  q1  <75  q3

N
N/2
N/2

good sequence

<2  q1

N

bad sequence
column with 100 unique integers $[1,100]$

- $<25$
- $<50$
- $<75$

**good sequence**

- N
- N/2
- N/2

**bad sequence**

- <2
- q1

Stochastic Cracking, PVLDB 12
column with 100 unique integers [1,100]

<25  <50  <75
q2   q1   q3

good sequence

<2  <3
q1   q2

bad sequence

N  N/2  N/2

Stochastic Cracking, PVLDB 12
column with 100 unique integers \([1, 100]\)

- Good sequence:
  - \(<25\): \(q_2\)
  - \(<50\): \(q_1\)
  - \(<75\): \(q_3\)

- Bad sequence:
  - \(<2\): \(q_1\)
  - \(<3\): \(q_2\)

- \(N\)
- \(N - 1\)
- \(N/2\)
- \(N/2\)
column with 100 unique integers [1,100]

<25  <50  <75
q2  q1  q3

good sequence

<2  <3
q1  q2

bad sequence
column with 100 unique integers [1, 100]

good sequence

<25  <50  <75
q2   q1   q3

bad sequence

<2  <3  <4
q1   q2   q3

N  N/2  N/2

N  N-1
column with 100 unique integers \([1,100]\)

**good sequence**

- \(<25\)  \(|q_2|\)
- \(<50\)  \(|q_1|\)
- \(<75\)  \(|q_3|\)

**bad sequence**

- \(<2\)  \(|q_1|\)
- \(<3\)  \(|q_2|\)
- \(<4\)  \(|q_3|\)

- \(N\)
- \(N/2\)
- \(N/2\)

- \(N\)
- \(N-1\)
- \(N-2\)
blindly adapting to queries is not always a good idea

<2  <3  <4
q1  q2  q3

bad sequence

N
N-1
N-2
query driven
query driven

to be cracked

q
random
query driven

progressive cracking
query driven

progressive cracking
q1: <v1

to be cracked
Stratos Idreos

query driven

Stochastic Cracking, PVLDB 12

progressive cracking
q1: <v1

random
to be cracked

crack + filter <v1
query driven

progressive cracking
q1: <v1

swap

random

crack + filter <v1
query driven

progressive cracking
q1: <v1

swap

random
crack + filter <v1

scan + filter <v1

to be cracked
query driven

progressive cracking
q1: <v1
q2: <v2
query driven

progressive cracking
q1: \(<v1\)
q2: \(<v2\)

random

scan + filter \(<v2\)

to be cracked
Stratos Idreos

query driven

progressive cracking
q1: <v1
q2: <v2

swap

scan + filter <v2

crack + filter <v2

Stochastic Cracking, PVLDB 12
query driven

progressive cracking
q1: <v1
q2: <v2

random

swap

scan + filter <v2

crack + filter <v2
cracking on Skyserver (4TB)
(Sloan Digital Sky Survey, www.sdss.org)

cracking answers 160,000 queries
while full indexing is still half way creating one index
multi-core utilization
select \([a,b]\)
select \([a, b]\)
select [a, b]
select [a,b]
select \([a, b]\)
Multi-cores, SIGMOD 15

core 1  core 2  core 3  ...  core N
Multi-cores, SIGMOD 15
Multi-cores, SIGMOD 15

Stratos Idreos
Multi-cores, SIGMOD 15
Multi-cores, SIGMOD 15
problem: cores may be under utilized

goal: either fully utilize a core or shut it down
when there is an underutilized CPU, pin a thread to it and to do a cracking task
partitions size - access frequency - hit ratio

random works best
10^8 tuples - 10 attributes, random queries

![Graph showing performance comparison between adaptive indexing and holistic indexing.](image)

- **Total Response Time (sec)**
  - X-axis: Adaptive Indexing, Holistic Indexing
  - Y-axis: 0 to 200
  - Data points:
    - Adaptive indexing: 900, 90, 9, 1
    - Holistic indexing: 900, 90, 9, 1

- **Cumulative # Index Partitions**
  - X-axis: Query sequence (x100)
  - Y-axis: 0 to 3500
  - Lines:
    - Adaptive indexing: Red diamonds
    - Holistic indexing: Green triangles
good for updates, storage
but we need joins
NORMALIZED DATA

good for updates, storage
but we need joins

DENORMALIZED DATA

only fast scans
but expensive to create,
storage & updates
First award in ACM SIGMOD undergrad research competition

Adaptive denormalization

Normalized data
First award in ACM SIGMOD undergrad research competition

Normalized data

Possible denormalized space

Adaptive denormalization
first award in ACM SIGMOD undergrad research competition

adaptive denormalization

continuously physically reorganize data based on incoming query patterns (joins)

normalized data

possible denormalized space
first award in ACM SIGMOD undergrad research competition

normalized data

adaptive denormalization

continuously physically reorganize data based on incoming query patterns (joins)

denormalized fragments queries only need to fast scan

possible denormalized space
first award in ACM SIGMOD undergrad research competition

adaptive denormalization

continuously physically reorganize data based on incoming query patterns (joins)

denormalized fragments
queries only need to fast scan

possible denormalized space

normalized data
Stratos Idreos

Adaptive denormalization

Continuously physically reorganize data based on incoming query patterns (joins)

Normalized data

Denormalized fragments

Queries only need to fast scan

Possible denormalized space

First award in ACM SIGMOD undergrad research competition
adaptive denormalization

continuously physically reorganize data based on incoming query patterns (joins)

denormalized fragments
queries only need to fast scan

possible denormalized space

first award in ACM SIGMOD undergrad research competition
daily data

years

[IBMbigdata]

data* skills

years

[StratosGuess]

data system design, set-up, tune, use
data systems that are easy to design
(storage, data flow, algorithms, tuning, etc)
Design & Development
Roles: Architects & Developers

Set-up and Tuning
Roles: Database Administrators

1. System Design
   - Workloads, h/w expected properties (performance, throughput, energy, budget, ...)

2. Prototype Implementation

3. Full Implementation

4. Set-up & Tune

5. Auto-tuning during query processing

application specific workload, h/w & expected properties
say we need a system for workload X (data/access patterns):
    should we strip down a relational system?
    should we build up a key-value store or main-memory system?
    should we build something from scratch?
say we need a system for workload X (data/access patterns):
  should we strip down a relational system?
  should we build up a key-value store or main-memory system?
  should we build something from scratch?

say the workload (read/write ratio) shifts (e.g., due to app features):
  should we use a different data layout for base data - diff updates?
  should we use different indexing or no indexing?
say we need a system for workload X (data/access patterns):
  should we strip down a relational system?
  should we build up a key-value store or main-memory system?
  should we build something from scratch?

say the workload (read/write ratio) shifts (e.g., due to app features):
  should we use a different data layout for base data - diff updates?
  should we use different indexing or no indexing?

say we buy new hardware X (flash/memory):
  should we change the size of b-tree nodes?
  should we change the merging strategy in our LSM-tree?
say we need a system for workload X (data/access patterns):
should we strip down a relational system?
should we build up a key-value store or main-memory system?
should we build something from scratch?

say the workload (read/write ratio) shifts (e.g., due to app features):
should we use a different data layout for base data - diff updates?
should we use different indexing or no indexing?

say we buy new hardware X (flash/memory):
should we change the size of b-tree nodes?
should we change the merging strategy in our LSM-tree?

say we want to improve response time:
would it be beneficial if we would buy faster flash disks?
would it be beneficial if we buy more memory?
or just spend the same budget on improving software?
# of design options?
# of design options?

> 1 QUINTILLION

(and counting…)

Stratos Idreos
data systems design (and research) is kind of an art
(both in the good & in the bad sense)
data systems design (and research) is kind of an art
(both in the good & in the bad sense)

but can we keep up?

Stratos Idreos
1 easily utilize past concepts/designs
2. do not miss out on cool ideas and concepts

The log-structured merge-tree (LSM-tree)
P. O’Neil, E. Cheng, D. Gawlick, E, O’Neil
The log-structured merge-tree (LSM-tree)

2

do not miss out on cool ideas and concepts
self-designing data systems

data+queries+hardware

can we design new systems in weeks instead of years?
self-designing data systems

data + queries + hardware

data system

easy to design

adapt to environment

can we design new systems in weeks instead of years?
INTERACTIVE DATA SYSTEM DESIGN/TUNING/TESTING
How to model/abstract design decisions?

How much can happen automatically (auto-design)?
How to model/abstract design decisions?

How much can happen automatically (auto-design)?

move from design based on intuition & experience only to a more formal and systematic way to design systems
1. write/extend modules in a high level language (optimizations)

2. modules = storage/executation/data flow

3. try out >>1 designs (sets of modules)
can we design systems where no SQL is needed?

data systems that are easy to use

dbTouch

cidr2013/icde2014

show me something interesting

Queriosity

bigdata2015
data systems today
allow us to answer queries fast

data systems tomorrow
should allow us to find fast which queries to ask
a semester of quizzes and brainstorming

option between systems project (key-value store) & research with DASlab
(only for CS165 students or otherwise advanced students)
learn the concepts, not just the techniques
learn to adapt
it all starts with how we store the data
yes you can do research

The End
DASlab productions 2016

/*project deadline Dec 21
OH and labs continue until then*/