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

raw data  →  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
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

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

piece1:
A <= 10
Q1: select R.A from R where R.A > 10 and R.A < 14

piece1: A <= 10

piece2: 10 < A < 14
Q1: select R.A from R where R.A > 10 and R.A < 14

<table>
<thead>
<tr>
<th>column A</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

- **Piece 1:** $A \leq 10$
  - 4
  - 9
  - 2
  - 7
  - 1
  - 3
  - 8
  - 6

- **Piece 2:** $10 < A < 14$
  - 13
  - 12
  - 11

- **Piece 3:** $A \geq 14$
  - 16
  - 19
  - 14
Q1: select R.A from R where R.A>10 and R.A<14

piece1: A<=10

piece2: 10<A<14

piece3: A>=14

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

gain knowledge on how data is organized

result

piece1: A <= 10

piece2: 10 < A < 14

piece3: A >= 14

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

dynamically/on-the-fly within the select-operator
gain knowledge on how data is organized

<table>
<thead>
<tr>
<th>column A</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>piece1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &lt;= 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>piece2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 &lt; A &lt; 14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>piece3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &gt;= 14</td>
</tr>
</tbody>
</table>

result
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
<table>
<thead>
<tr>
<th>Q1: select R.A from R where R.A&gt;10 and R.A&lt;14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2: select R.A from R where R.A&gt;7 and R.A&lt;=16</td>
</tr>
</tbody>
</table>

column A

<table>
<thead>
<tr>
<th>A</th>
<th>A &lt;= 10</th>
<th>10 &lt; A &lt; 14</th>
<th>A &gt;= 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

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

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

piece1: A <= 10
piece2: 10 < A <= 14
piece3: A >= 14
Q1:
select R.A from R
where R.A > 10
and R.A < 14

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

Q2:
select R.A from R
where R.A > 7
and R.A <= 16
Q1:
select R.A
from R
where R.A>10
and R.A<14

column A

Q2:
select R.A
from R
where R.A>7
and R.A<=16
dynamically/on-the-fly within the select-operator

piece1: A<=7
piece2: 7<A<=10
piece3: 10<A<14
piece4: 14<=A<=16
piece5: A>16

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

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

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

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

almost no
initialization overhead

continuous adaptation

Database Cracking CIDR 2007
set-up
100K random selections
random selectivity
random value ranges
in a 10 million integer column

almost no
initialization overhead

continuous improvement

Database Cracking CIDR 2007
set-up

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

almost no
initialization overhead

continuous improvement

Database Cracking CIDR 2007
cracking databases

- basics (CIDR07)
- updates (SIGMOD07)
- >1 columns (SIGMOD09)
- storage-restrictions (SIGMOD09)
- algorithms (PVLDB11)
- benchmarking (TPCTC10)
- encryption (SIGMOD16)
- multi-cores (SIGMOD15)
- time-series (SIGMOD14)
- adaptive storage (SIGMOD14)
- hadoop (Yale/Saarland)
- b-trees (HP Labs)
- concurrency control (PVLDB12)

- robustness (PVLDB12)
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
traditional indexing

change index contents and structure

write queries

adaptive indexing

only index structure changes

read queries

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

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

- **change index contents and structure**
  - traditional indexing
  - **write queries**
- **only index structure changes**
  - adaptive indexing
  - **read queries**
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
Stratos Idreos

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
**Stratos Idreos**

**Concurrency Control, PVLDB 12**

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

**Adaptive Indexing**
- Incremental and optional
- Only index structure changes
- Read queries
Stratos Idreos

traditional indexing

all or nothing

incremental and optional

change index contents and structure

only index structure changes

write queries

read queries

Concurrency Control, PVLDB 12
Stratos Idreos

Concurrencies Control, PVLDB 12
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
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
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20

disk

memory

A B C D

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

disk

memory

A  B  C  D

A<10  IDs  B
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

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  crack column
piece locking

avl-tree  crack column

wlock

crack select
Concurrent locking

Concurrent locking
piece locking

avl-tree  crack column
piece locking

avl-tree  crack column

max
piece locking

avl-tree

crack column

rlock

max
Stratos Idreos

Concurrency Control, PVLDB 12

piece locking

avl-tree

crack column

max

rlock
Concurrency Control, PVLDB 12

Stratos Idreos
piece locking

avl-tree  crack column

10  
90  
140  
200  
300  

Stratos Idreos
piece locking

avl-tree  crack column  crack select

10  90  140  200  300

65  230
piece locking

avl-tree

10
90
140
200
300

wlock

65

230

crack select
piece locking

avl-tree  crack column  crack select

r/wlock

wlock

wlock

10
90
140
200
300

65
230

Stratos Idreos
Stratos Idreos

avl-tree

crack column

wlock

q1, q2, q3, ..., qn

crack select

piece locking

wlock

65

230

10

90

140

200

300
Stratos Idreos

avl-tree

crack column

wlock

q1, q2, q3, ..., qn

piece locking

10

90

140

200

300

65

230

10

20

30

70

80

90
Stratos Idreos

avl-tree

piece locking

crack column

wlock

q_1, q_2, q_3, \ldots, q_n

wlock

wlock

10

90

140

200

300

10

20

30

70

80

230

65
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

Concurrency Control

(Sequential Execution)
Crack time and wait time

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

Concurrency Control, PVLDB 12
stochastic cracking
robustness
adaptive indexing
Response time: X

adaptive indexing
Response time: X
Response time: $X$

Response time: $1000X$
v5   v4   v3   v2   v1

v1   v2   v3   v4   v5
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]
Stratos Idreos

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]
Good sequence column with 100 unique integers [1,100]
column with 100 unique integers [1, 100]
<50
q1
good sequence
column with 100 unique integers [1,100]

<50
q1

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

<50

\(q_1\)

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

<25  <50

q2    q1

good sequence

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

<25
q2

<50
q1

<75
q3

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

$q_2 < 25$
$q_1 < 50$
$q_3 < 75$

N
N/2
N/2

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

<25  q2  <50  q1  <75  q3

good sequence

bad sequence

<2  q1

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

<25
q2
<50
q1
<75
q3

good sequence

<2
q1

bad sequence

N
N/2
N/2

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

<25
q2
<50
q1
<75
q3

good sequence

<2
q1

bad 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

<2  <3
q1   q2

bad sequence

N
column with 100 unique integers [1, 100]

 buena secuencia

<25 q2 <50 q1 <75 q3

N
N/2

N-1

Mal secuencia
column with 100 unique integers [1, 100]

good sequence

<25

<50

<75

bad sequence

<2

<3

<50

<25

<75

N

N/2

N/2

N

N-1
column with 100 unique integers [1,100]

- good sequence
  - $<25$ (q2)
  - $<50$ (q1)
  - $<75$ (q3)

- bad sequence
  - $<2$ (q1)
  - $<3$ (q2)
  - $<4$ (q3)

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

<25  <50  <75
q2   q1   q3

good sequence

<2  <3  <4
q1   q2   q3

bad sequence
blindly adapting to queries is not always a good idea
query driven
to be cracked

Stochastic Cracking, PVLDB 12
query driven

to be cracked

q random
query driven

progressive cracking

Stochastic Cracking, PVLDB 12
progressive cracking
q1: \(<v1

query driven

Stochastic Cracking, PVLDB 12
progressive cracking
q1: \(<v1

\text{crack + filter } <v1

to be cracked
query driven

progressive cracking
q1: <v1

swap

random

crack + filter <v1

to be cracked
Stratos Idreos

query driven

progressive cracking
q1: <v1

random

swap

crack + filter <v1

scan + filter <v1

to be cracked
progressive cracking
q1: <v1
q2: <v2

query driven

Stochastic Cracking, PVLDB 12
query driven

progressive cracking

q1: <v1
q2: <v2

random

scan + filter <v2
progressive cracking
q1: <v1
q2: <v2

scan + filter <v2

Stochastic Cracking, PVLDB 12

query driven
query driven

progressive cracking
q1: <v1
q2: <v2

swap

random

crack + filter <v2

scan + 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]

< a

>= a

Multi-cores, SIGMOD 15
select [a,b]

$\langle a \rangle \geq a$

multi-cores, SIGMOD 15
Multi-cores, SIGMOD 15
Multi-cores, SIGMOD 15
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](image)

**Figure 6(a)** shows the results. On the x-axis represents the cumulative response time of the query sequence, and on the y-axis represents the total response time (in seconds). The graph demonstrates the performance comparison between adaptive indexing and holistic indexing. The adaptive indexing shows a higher response time initially but improves as the query sequence grows, whereas the holistic indexing shows a more linear response time.

The graph on the right illustrates the cumulative number of index partitions generated over the query sequence. The y-axis represents the cumulative # index partitions, and the x-axis represents the query sequence (in units of 100). The graph compares adaptive indexing (red dotted line) and holistic indexing (green line) with a linear increase in index partitions as the query sequence progresses.

In summary, the holistic indexing approach leads to a more efficient performance improvement over adaptive indexing, especially in dynamic and ad-hoc environments. The holistic approach also demonstrates better stability against parallelization and can be applied independently without incurring additional costs for workload knowledge acquisition.
a semester of quizzes and brainstorming

option between 2 small projects
& research with DASlab

(only for CS165 students or otherwise advanced students)
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)

data has always been big

volume  velocity  variety  veracity

Stratos Idreos
Stratos Idreos

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)
e.g., column-stores/main-memory optimized systems:
first ideas in 80s,
first advanced architectures in 90s,
first rather complete designs in early 2000s,
mainstream industry adoption 2010+
still no indexing, limited cost based optimizations, …
conflicting goals
(hardware and requirements change continuously and rapidly)

moving target

application requirements

hardware

performance

budget

energy profile

Stratos Idreos
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?
1. write/extend modules in a high level language (optimizations)

2. modules = storage/execution/data flow

3. try out >1 designs (sets of modules)

GENOME-Synthesizer

Gene pool (reuseable modules)

Custom tuned architecture

Monitor & detect workload changes

row-store

hybrid store

key-value store

column-store

Workload & feature list

Hardware description
can we design systems where no SQL is needed?

data systems that are easy to use

dbTouch

cidr2013/icde2014

data that is easy to use

show me something interesting

Queriosity

bigdata2015

--- DATA

Stratos Idreos

Harvard School of Engineering and Applied Sciences
data systems today
allow us to answer queries fast

data systems tomorrow
should allow us to find fast which queries to ask
Overview of Data Exploration Techniques
S. Idreos, O. Papaemmanouil, and S. Chaudhuri
ACM SIGMOD International Conference on Management of Data, 2015

(explore citations)
The End
DASlab productions 2015

\*project deadline Dec 21
OH continue until then*/