auto-tuning database kernels

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
now you know the basics!

and it is time for research

we are going to go through the process of designing a full data system kernel from scratch
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
not always sure what we are looking for (until we find it)

data has always been big

volume    velocity    variety    veracity
Stratos Idreos

years

[IBMbigdata]

data system design, set-up, tune, use

[StratosGuess]
data systems that are easy to:

design & build (years)

set-up & tune (months)

use (hours/days)
+ of course make things fast
today and Wednesday

adaptive indexing, adaptive loading, adaptive storage, dbTouch

since 2005

joint work with:

Martin Kersten
Stefan Manegold
Goetz Graefe
Harumi Kuno
Anastasia Ailamaki
Eleni Petraki

Ioannis Alagiannis
Miguel Branco
Renata Borovica
Felix Halim
Ronald Yap
Panos Karras
too many preparation options lead to complex installation

expert users - idle time - workload knowledge
users/applications
declarative interface
ask what you want

db system

DBA
need to choose the proper system & workloads/applications change rapidly
be able to query the data immediately & with good performance
be able to query the data immediately & with good performance

raw data  explore data and gain knowledge “immediately”
tune= create proper indices offline
performance 10-100X
tune = create proper indices offline
performance 10-100X

but it depends on workload!
which indices to build?
on which data parts?
and when to build them?
storage  load  indexing  query
storage  load  indexing  query
sample workload
storage load indexing query

sample workload analyze

timeline
storage  load  indexing  query

sample workload  analyze  create indices

timeline
sample workload  analyze  create indices  query

timeline

complex and time consuming process
human administrators + auto-tuning tools

dsampling workload → analyze → create indices → query

timeline

complex and time consuming process
what can go wrong?

- **not enough space** to index all data
- **not enough idle time** to finish proper tuning
- **by the time we finish tuning, the workload changes**
- **not enough money** - energy - resources
what can go wrong?

- not enough space to index all data
- not enough idle time to finish proper tuning
- by the time we finish tuning, the workload changes
- not enough money - energy - resources
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

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

every query is treated as an advice on how data should be stored
column-store database
a fixed-width and dense array per attribute
<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>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>11</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

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>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>3</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

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

sort

binary search
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

piece1: A <= 10

piece2: 10 < 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

result

gain knowledge on how data is organized
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
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

column A

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 13| 16| 12| 7| 1| 9| 4| 2| 12| 7| 1| 9| 4| 2| 12| 7| 1| 9| 4| 2| 12| 7| 1| 9| 4| 2| 12| 7| 1| 9| 4| 2|

piece1: A <= 10

piece2: 10 < A <= 14

piece3: A >= 14

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
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

result
<table>
<thead>
<tr>
<th>column A</th>
<th>件1: A&lt;=10</th>
<th>件2: 10&lt;A&lt;=14</th>
<th>件3: A&gt;=14</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>4</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>9</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>14</td>
<td>1</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

the more we crack, the more we learn

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

Database Cracking CIDR 2007
select [15,55]
select [15,55]
select [15,55]

10  20  30  40  50  60

select [15,55]
touch at most two pieces at a time

pieces become smaller and smaller

select [15,55]
set-up

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

Response time (secs)

Query sequence (x1000)

0.001
0.01
0.1
1
10
100
1000
10000
100000

Scan
Crack
Full Index

Database Cracking CIDR 2007
set-up

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

almost no initialization overhead
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
set-up

10K random selections
selectivity 10%
random value ranges
in a 30 million integer column

Continuous average response time (secs)

Query sequence

Full Index
Scan
Crack

Database Cracking CIDR 2007
set-up

10K random selections
selectivity 10%
random value ranges
in a 30 million integer column

10K queries later,
Full Index still has not
amortized the initialization costs
<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**table1**
select R.A from R where R.A > 10 and R.A < 14
select R.A from R where R.A > 10 and R.A < 14

select max(R.A), max(R.B), max(S.A), max(S.B) from R, S
where v1 < R.C < v2 and v3 < R.D < v4
and v5 < R.E < v6 and k1 < S.C < k2 and k3 < S.D < k4 and k5 < S.E < k6
and R.F = S.F
select R.A from R where R.A>10 and R.A<14

select max(R.A), max(R.B), max(S.A), max(S.B) from R,S
where v1 < R.C < v2 and v3 < R.D < v4
and v5 < R.E < v6 and k1 < S.C < k2 and k3 < S.D < k4 and k5 < S.E < k6
and R.F = S.F
cracking databases

- basics (CIDR07)
- updates (SIGMOD07)
- >1 columns (SIGMOD09)
- storage-restrictions (SIGMOD09)
- algorithms (PVLDB11)
- multi-cores (SIGMOD15)
- encryption (SIGMOD16)
- hadoop (Yale/Saarland)
- b-trees (HP Labs)
- concurrency control (PVLDB12)
- robustness (PVLDB12)
- time-series (SIGMOD14)
- adaptive storage (SIGMOD14)
- benchmarking (TPCTC10)
- encryption (SIGMOD16)
updates
cracking indices are auxiliary data structures can be dropped any time
Forgetting

When updates arrive, drop the index

crack (forget)

scan

Stratos Idreos
column of 10M tuples, random queries
1000 random insertions every 1000 queries

we do not exploit past cracking
ing when updates arrive, drop the index

we do not exploit past cracking
log updates and apply on-line and on-demand during cracking
log updates and apply on-line and on-demand during cracking

updates

query

maintain
goal: minimize physical actions

arrays are dense

pieces are ordered

values in a piece are not ordered

3
1
9
10
17
12
16
20
25
22
29
30
33
39
31

pending inserts
pending deletes
goal: minimize physical actions

arrays are dense

pieces are ordered

values in a piece are not ordered

---

column

10

11

20

30

pending inserts

pending deletes
goal: minimize physical actions

arrays are dense

pieces are ordered

values in a piece are not ordered

column

10

11

20

22

25

29

30

31

33

pending inserts

pending deletes
goal: minimize physical actions

arrays are dense

pieces are ordered

values in a piece are not ordered

pending inserts

pending deletes
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered

column

10

11

20

30

33

pending inserts

pending deletes
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered

column

10
3
1
9

11
17
12
16

20

22
29
25

30
39
31
33

pending inserts

pending deletes
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered

column

values:

pending inserts
pending deletes
column of 10M tuples, random queries
1000 random insertions every 1000 queries

merge all updates
merge only qualifying updates

queries (x1000)
queries (x1000)
column of 10M tuples, random queries
1000 random insertions every 1000 queries

merge all updates
merge only qualifying updates

ggradual merging avoids high picks
the ripple

query result

pending updates

column
the ripple
the ripple

query result

column

pending updates
the ripple

query result

column

doing work not relevant for the current query

pending updates
the ripple

doing work not relevant for the current query
the ripple

pending updates

query result
doing work not relevant for the current query
the ripple

doing work not relevant for the current query
pending updates

doing work not relevant for the current query
query result

pending updates

doing work not relevant for the current query

query result
The ripple column pending updates doing work not relevant for the current query.
the ripple

doing work not relevant for the current query
Stratos Idreos

the ripple

column

pending updates

query result

doing work not relevant for the current query

query result

column

query result
the ripple

query result

column

pending updates

doing work not relevant for the current query

column

query result
the ripple

query result

column

pending updates

column

update only the hot tuples

doing work not relevant for the current query
Stratos Idreos

Forget: maintain adaptive behavior
Ripple: merge only qualifying updates

Cost per query (microseconds)

Queries (x1000)
sideways cracking
how to process >1 columns
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a1</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>2</td>
<td>a2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>3</td>
<td>a3</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>4</td>
<td>a4</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>5</td>
<td>a5</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>6</td>
<td>a6</td>
<td>b6</td>
<td>c6</td>
</tr>
</tbody>
</table>

positional alignment

positional lookups

$$A(i) = A + i \times \text{width}(A)$$
select min(C) from R where A<10 & B<20
\textbf{select} max(D),min(E) \textbf{from} R \textbf{where} (A>10 \text{ and } A<40) \text{ and } (B>20 \text{ and } B<60)

\textbf{avoid scan} of A \\
\textbf{avoid TR} on B \\
work on a \textbf{restricted area} \\
across all columns \\
good for memory hierarchy
without cracking
ordered
sequential access

with cracking
unordered
random access
sideways cracking
sideways cracking
Sideways Cracking

A

B

C

D

sideways cracking
query

sideways cracking

A
B
C
D
query

A

B

C

D

sideways cracking
Sideways Cracking, SIGMOD 09

query

A

B

C

D

Stratos Idreos
Sideways Cracking, SIGMOD 09
Sideways Cracking, SIGMOD 09

query

A

B

C

D


sideways cracking

query

Stratos Idreos

Sideways Cracking, SIGMOD 09
sideways cracking

A

B

C

D

query

Sideways Cracking, SIGMOD 09
Sideways Cracking, SIGMOD 09

A

B

C

D

query
Sideways Cracking, SIGMOD 09
log crack actions and replay to align columns dynamically
replace tuple reconstruction with cracking actions
### Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>
sideways cracking

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

Initial state | select B from R where 10<A<15

Position 1 | value <=10
Piece 1

Position 7 | value >10
Piece 2

Position 9 | value >=15
Piece 3

Position 4 | value >=5
Piece 2

Position 1 | value >=17
Piece 5
## Sideways Cracking

**Initial state**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

**Select B from R where 10 < A < 15**

<table>
<thead>
<tr>
<th>Cracker index</th>
<th>M_{AB}</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

**Cracker index:**

- **Piece 1:** Position 1, value ≤ 10
- **Piece 2:** Position 7, value > 10
- **Piece 3:** Position 9, value ≥ 15
Sideways Cracking, SIGMOD 09

Stratos Idreos
sideways cracking

Cracker Map

<table>
<thead>
<tr>
<th>Initial state</th>
<th>select B from R where 10&lt;A&lt;15</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

Cracker index

<table>
<thead>
<tr>
<th>Cracker index</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_{AB}</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>16</td>
</tr>
</tbody>
</table>

Piece 1

<table>
<thead>
<tr>
<th>Piece 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
</tr>
<tr>
<td>Position 1</td>
</tr>
<tr>
<td>value &lt;=10</td>
</tr>
<tr>
<td>Piece 2</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>Position 7</td>
</tr>
<tr>
<td>value &gt;10</td>
</tr>
<tr>
<td>Piece 3</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>Position 9</td>
</tr>
<tr>
<td>value &gt;=15</td>
</tr>
</tbody>
</table>

Crack based on head, carry tail

Sideways Cracking, SIGMOD 09
Sideways Cracking, SIGMOD 09

Cracker Map

- Head
- Tail

Select B from R where 10 < A < 15

Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

Cracker index

1. Position 1: value ≤ 10
   - Piece 1
   - b1

2. Position 7: value > 10
   - Piece 2
   - b8

3. Position 9: value ≥ 15
   - Piece 3
   - b13

Cracking knowledge

- Crack based on head, carry tail.

stratos idreos
Sideways Cracking, SIGMOD 09

Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

select B from R where 10<A<15

Cracker index

<table>
<thead>
<tr>
<th>M_{AB}</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>16</td>
</tr>
</tbody>
</table>

Cracking knowledge

Cracker Map

Head

Tail

Crack based on head, carry tail

No tuple reconstruction

Stratos Idreos
Sideways Cracking, SIGMOD 09

Cracker Map

Crack based on head, carry tail

Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

select B from R where 10 < A < 15

Cracker index

<table>
<thead>
<tr>
<th>M_{AB}</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

Crack based on head, carry tail

Piece 1

Position 1
value <= 10

Piece 2

Position 7
value > 10

Piece 3

Position 9
value >= 15

Cracking knowledge
No tuple reconstruction

Dynamically/on-the-fly within the select-operator
sideways cracking

Cracker Map
Head  Tail

Cracker index

select B from R where 10<A<15

<table>
<thead>
<tr>
<th>Initial state</th>
<th>select B from R where 10&lt;A&lt;15</th>
<th>select B from R where 5=&lt;A&lt;17</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>Cracker index</td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
<td>12</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
<td>22</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
<td>24</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
<td>16</td>
</tr>
</tbody>
</table>

Crack based on head, carry tail

Cracking knowledge
No tuple reconstruction
Dynamically/on-the-fly within the select-operator

Stratos Idreos

Sideways Cracking, SIGMOD 09
Sideways Cracking, SIGMOD 09
Stratos Idreos

Sideways Cracking, SIGMOD 09
perform the same cracks and in the same order on all maps with the same head
perform the same cracks and in the same order on all maps with the same head

on-line alignment touch/load everything, always
perform the same cracks and in the same order on all maps with the same head

on-line alignment touch/load everything, always
perform the same cracks and in the same order on all maps with the same head on-line alignment touch/load everything, always

remember and replay cracks across columns
adaptive alignment

Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>4</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>2</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c7</td>
</tr>
</tbody>
</table>

select B from R where A<3

Crack A<3

M_{AB}

Result

v<3

b4

b3

v>=3

select C from R where A<5

Crack A<5

M_{AC}

Result

v<5

c6

c2

c7

Wrong alignment

select B,C from R where A<4

Crack A<4

M_{AB}

Result

v<3

b4

b1

v>=3

b3

select C from R where A<5

Crack A<5

M_{AC}

Result

v<5

c6

c2

c7

Correct alignment

Stratos Idreos

Sideways Cracking, SIGMOD 09
Sideways Cracking, SIGMOD 09
Stratos Idreos

Sideways Cracking, SIGMOD 09
adapted alignment

replace tuple reconstruction with cracking

Wrong alignment

Correct alignment

Sideways Cracking, SIGMOD 09
wider maps...but too many combinations
maps of different maps sets lead to alignment problems

select D from R where 3<A<10 and 4<B<8 and 1<C<7
multi-selections

wider maps...but too many combinations
maps of different maps sets lead to alignment problems
use a single map set and exploit bit-vectors

select D from R where 3<A<10 and 4<B<8 and 1<C<7
select D from R where 3<A<10 and 4<B<8 and 1<C<7
select D from R where 3<A<10 and 4<B<8 and 1<C<7

Crack 3<A<10
Analyze tail 4<B<8
Create bit vector
select D from R where 3<A<10 and 4<B<8 and 1<C<7

Crack 3<A<10
Analyze tail 4<B<8
Create bit vector

Align 3<A<10
Analyze tail 1<C<7
Refine bit vector
select D from R where 3<A<10 and 4<B<8 and 1<C<7

Crack 3<A<10
Analyze tail 4<B<8
Create bit vector

Align 3<A<10
Analyze tail 1<C<7
Refine bit vector

Align 3<A<10
Grab tail values
select D from R where 3<A<10 and 4<B<8 and 1<C<7

Crack 3<A<10
Analyze tail 4<B<8
Create bit vector

Align 3<A<10
Analyze tail 1<C<7
Refine bit vector

Align 3<A<10
Grab tail values

Use histogram-like info from maps to choose map set
TPC-H Query 15

- MonetDB
- Sel. Crack
- MySQL
- Presorted
- Sid. Crack
- Presorted

Response time (milli secs)

Query sequence

normal MonetDB

selection cracking

Sideways Cracking, SIGMOD 09
presorted MonetDB preparation cost 3-14 minutes per query

Response time (milli secs)

Query sequence

TPC-H Query 15

MonetDB
Presorted
Sel. Crack
Sid. Crack
MySQL
Presorted

normal MonetDB
selection cracking
presorted MonetDB preparation cost 3-14 minutes per query
presorted MonetDB preparation cost 3-14 minutes per query

Response time (milli secs)

Query sequence

TPC-H Query 15

Sideways Cracking, SIGMOD 09
presorted MonetDB preparation cost 3-14 minutes per query

MonetDB
Presorted
Sel. Crack
Sid. Crack
MySQL
Presorted

Response time (milli secs)

Query sequence

TPC-H Query 15

Sideways Cracking, SIGMOD 09
presorted MonetDB preparation cost 3-14 minutes per query

MonetDB with sideways cracking

selection cracking

normal MonetDB

TPC-H Query 15

3-14 minutes per query
Partial Sideways Cracking

**Initial state**

<table>
<thead>
<tr>
<th>Keys</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>b7</td>
<td>c7</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>b8</td>
<td>c8</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>b9</td>
<td>c9</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>b10</td>
<td>c10</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>b11</td>
<td>c11</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>b12</td>
<td>c12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>b13</td>
<td>c13</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>b14</td>
<td>c14</td>
</tr>
</tbody>
</table>

**Select B from R where 9<A<=15**

**Chunk map $H_A(A,id)$**

- **Partial map $M_{AB}$**
  - $v<9$: $E$
  - $v>9$: $M,C=1$
  - $v>15$: $E$

**Select B from R where 9<A<15**

**Chunk map $H_A(A,id)$**

- **Partial map $M_{AB}$**
  - $v<9$: $E$
  - $v>9$: $M,C=1$
  - $v>15$: $E$

**Select B from R where 5<=A<8**

**Chunk map $H_A(A,id)$**

- **Partial map $M_{AB}$**
  - $v<5$: $E$
  - $v=5$: $M,C=2$
  - $v>=5$: $E$
  - $v>9$: $M,C=1$
  - $v>8$: $E$
  - $v>9$: $M,C=1$
  - $v>15$: $E$

**Select C from R where 8<=A<15**

**Chunk map $H_A(A,id)$**

- **Partial map $M_{AC}$**
  - $v<8$: $E$
  - $v=8$: $M,C=3$
  - $v>=8$: $E$
  - $v>9$: $M,C=1$
  - $v>13$: $E$
  - $v>15$: $E$

---

**Keys**

1  2  3  4  5  6  7  8  9  10  11  12  13  14

**Select B from R where 9<A<=15**

**Partial map $M_{AB}$**

- $v<=9$: $E$
- $v>9$: $M,C=1$
- $v>15$: $E$

**Select C from R where 8<=A<15**

**Partial map $M_{AC}$**

- $v<=9$: $E$
- $v>9$: $M,C=1$
- $v>15$: $E$
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
Change index contents and structure

Write queries

Traditional indexing

Only index structure changes

Read queries

Adaptive indexing
no need for traditional locks = too heavy

short term latches = fast and release quickly

change index contents and structure
write queries
traditional indexing

only index structure changes
read queries
adaptive indexing
traditional indexing

all or nothing
change index contents and structure
write queries

traditional indexing

incremental and optional
only index structure changes
read queries
all or nothing
incremental and optional
change index contents and structure
only index structure changes
write queries
read queries
traditional indexing
adaptive indexing
Stratos Idreos

Concurrency Control, PVLDB 12

all or nothing

incremental and optional

change index contents and structure

only index structure changes

write queries

read queries

traditional indexing

adaptive indexing
Stratos Idreos

traditional indexing

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

**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

Concurrent Control, PVLDB 12

incremental and optional

only index structure changes

read queries

adaptive indexing
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
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
traditional indexing

impact stable storage

all or nothing

change index contents and structure

write queries

need to serialize

can execute in any order

stable storage optional

incremental and optional

only index structure changes

read queries

adaptive indexing

Concurrency Control, PVLDB 12
fewer conflicts as we adapt
late reconstruction

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

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

late reconstruction
late reconstruction

\[ \text{select min}(C) \text{ from } R \text{ where } A < 10 \text{ } \& \text{ } B < 20 \]

disk

memory

A  B  C  D

A < 10  IDs

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

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

disk

memory

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

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

disk

memory

A B C D

A<10 IDs B B<20 IDs C minC

column lock and release as soon as an operator completes
select [a,b]
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
piece locking

avl-tree  crack column  crack select

wlock
piece locking

 AVL-Tree → Crack Column
piece locking

avl-tree

crack column

max
Concurrency Control, PVLDB 12
piece locking

avl-tree crack column

max

rlock
Concurrency Control, PVLDB 12
piece locking

avl-tree crack column

max

clock

Concurrent Control, PVLDB 12
piece locking
Stratos Idreos

piece locking

avl-tree

crack column

10
90
140
200
300
65
230

crack select
piece locking

avl-tree  crack column  crack select

wlock

10  90  140  200  300

65  230
Stratos Idreos

piece locking

avl-tree  crack column  crack select

r/wlock

wlock

10  90  90  140  200  300

65  230
piece locking

avl-tree

crack column

wlock

crack select

q1, q2, q3...qn

wlock

230

65
Stratos Idreos

avl-tree

crack column

wlock

q1, q2, q3... qn

wlock

piece locking
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

Concurrent Execution

(Sequential Execution)
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

adaptive behavior also for conflicts
Overview of Data Exploration Techniques
S. Idreos, O. Papaemmanouil, and S. Chaudhuri
ACM SIGMOD International Conference on Management of Data, 2015

(explore citations)
auto-tuning database kernels

DATA SYSTEMS

prof. Stratos Idreos