indexing & sorting

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
first part done: basic concepts in modern systems

coming up: indexing and fast scans
it all starts with how we layout the data (bits)
essential column-stores features
- virtual ids
- late tuple reconstruction (if ever)
- vectorized execution
- compression
- fixed-width columns

Column-stores vs. row-stores: how different are they really?
D. Abadi, S. Madden, and N. Hachem
ACM SIGMOD Conference on Management of Data, 2008
Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD  Edgar F. Codd Innovations award

100Kx disk
Pluto
2 years

100x memory
New York
1.5 hours

10x on board cache
this building
10 min

2x on chip cache
this room
1 min

registers
my head
~0
random access & page-based access

data value $x$... but have to read all of page 1

CPU
registers
on chip cache
on board cache
memory
disk
and then from ideas to practice: implementation, tuning, analysis
as you are starting your projects, remember

come to office hours - read/post in piazza

distributed API, DSL, code is supposed to help you start fast
diverging is perfectly ok when there is a good reason to do so

**functionality goal:**
```
select max(R.a), min(S.a) from R, S
where R.j=S.j and R.b<20 and S.c>10 and S.d<50
```
+updates and persistency

**performance goal:**
scability (cores/queries)
cache conscious
vectorwised processing: how to

select max(A) from R where B<20

\[ p = \text{select}(B, \text{null}, 20) \]
\[ a = \text{fetch}(A, p) \]
\[ \text{res} = \text{max}(a) \]

\[ \text{j} = 0; \]
\[ \text{for} (i = 0; i < B.\text{size}; i + \text{vector.}\text{size}) \{
    p = \text{select}(B, i, \text{vector.}\text{size}, \text{null}, 20)
    a = \text{fetch}(A, p)
    \text{rv}[\text{j++}] = \text{max}(a)
}\]
\[ \text{res} = \text{max}(\text{rv}) \]

rewritten to

assume optimizer does the rewriting and focus on analysis of property X - vectorwised vs column-at-a-time

Enhanced stream processing in a DBMS kernel
Erietta Liarou, Stratos Idreos, Stefan Manegold, Martin Kersten
In Proc. of the International Conf. on Extending Database Technology, 2013
10/12

**midterms**

how to prepare

- open book, notes, no laptop/discussion
- material from lectures only
- check all quizzes and questions

**quiz-like questions - no exact answer**

*explain all steps and tradeoffs*

**expectations:** describe the design space - chose what you think is the best approach (>1 if we ask for it) and then analyze in detail all requests - if you made the wrong choice in the begging it is ok - but say so if you find out in the end and explain as much as possible

**Sunday before midterm:** Stratos office hours 2pm+
today+3

data access made better
it all starts with the select operator

it touches all the data
index knows structure of the data
filtering data: point/range queries

an alternative data representation
(data structure) of all or part of the data
but wait, why not just sort the data (array) + binary search?

why bother with creating/maintaining another data structure?
ok let’s go with sorting for a while

initial state
columns in insertion order

sorted

select B+C from R
where A<10
values are out of order
values are out of order
values are out of order

intermediate out of order
sort or cluster
applications

data

data

algorithms/operators

database kernel

sql

data

data

data

cpu

memory

disk

CS165, Fall 2016
Stratos Idreos
initial state
columns in
insertion order

sorted A  B  C

propagate
order of A
\textbf{select} \textbf{max}(D), \textbf{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
**select** \( \text{max}(D), \text{min}(E) \) **from** \( R \) **where** \((A>10 \text{ and } A<40) \text{ or } (B>20 \text{ and } B<60)\)
C-Store: A Column-oriented DBMS
Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alex Rasin, Nga Tran, Stanley B. Zdonik
In Proc. of the Very Large Databases Conference (VLDB), 2005
initial state
columns in
insertion order

base data

space overhead - update overhead - which ones to build?
declarative interface
ask what you want

indexes/views/tuning knobs

db system
online

storage budget $\ll$ smaller than the possible set of projections
cost to sort array $Cs$?
cost to find a value once sorted $Ca$?
on optimized algorithm to minimize $Cs$ & $Ca$

data does not fit in level 1 memory
CPU can read/write directly from/to level 1 only

(assume simplified memory hierarchy)
initial state: 8 unordered pages
quicksort in place

memory level L

memory level L-1

(initial state: 8 unordered pages)
memory level L

memory level L-1

(initial state: 8 unordered pages)
quicksort in place

memory level L

(initial state: 8 unordered pages)

memory level L-1

(size=3 pages)
memory level L

memory level L-1

(initial state: 8 unordered pages)
quicksort in place (size=3 pages)

initial state: 8 unordered pages
each page is now sorted
we read and wrote every page once
data movement cost is $2N$ pages

initial state: 8 unordered pages
initial state: 8 sorted pages
merge to new page

memory level L

memory level L-1

(initial state: 8 sorted pages)
merge to new page

memory level L

memory level L-1

(initial state: 8 sorted pages)

(size=3 pages)
initial state: 8 sorted pages

(merge to new page)

memory level L

memory level L-1

(size=3 pages)
merge to new page

memory level L

(memory level L-1)

(initial state: 8 sorted pages)
merge to new page

memory level L

memory level L-1

(initial state: 8 sorted pages)
initial state: 8 sorted pages
Each pair of pages is now sorted. We read and wrote every page once.

Data movement cost is $2N$ pages (total $2N + 2N$)

Initial state: 8 sorted pages
1 pass to sort each page (2N pages)
1 pass to sort each page (2N pages)

1 pass to merge into 2 sorted pages (2N pages)

1 pass to merge into 4 sorted pages (2N pages)

1 pass to merge into 8 sorted pages (2N pages)
1 pass to sort each page (2N pages)

1 pass to merge into 2 sorted pages (2N pages)

1 pass to merge into 4 sorted pages (2N pages)

1 pass to merge into 8 sorted pages (2N pages)

$\log_2(N)$
1 pass to sort each page (2N pages)

1 pass to merge into 8 sorted pages (2N pages)

1 pass to merge into 4 sorted pages (2N pages)

1 pass to merge into 2 sorted pages (2N pages)

\[ \log_2(N) + 1 \]
1 pass to sort each page (2N pages)

1 pass to merge into 2 sorted pages (2N pages)

1 pass to merge into 4 sorted pages (2N pages)

1 pass to merge into 8 sorted pages (2N pages)

$2N(\log_2(N) + 1)$
1 pass to sort each page (2N pages)

1 pass to merge into 2 sorted pages (2N pages)

1 pass to merge into 4 sorted pages (2N pages)

1 pass to merge into 8 sorted pages (2N pages)

\[2N(\log_2(N)+1)\times \text{bytesPerPage}\]
in general, we have M pages in memory not just 3 so &

in our first pass we can immediately sort groups of M pages &
in general, we have \( M \) pages in memory not just 3 so

\[
2N(\log_2(N)+1) \rightarrow 2N(\log_{M-1}(N)+1)
\]

&

in our first pass we can immediately sort groups of \( M \) pages

\[
2N(\log_{M-1}(N)+1) \rightarrow 2N(\log_{M-1}(N/M)+1)
\]
data size: N pages
memory size: M pages

how much memory M do we need to sort N data in p passes only?

or

how much data can we sort in p passes if we have M memory?

\[ \log_{M-1}(N/M) + 1 \leq p \]
previous discussion holds for all levels of memory hierarchy
other usage of sorting, e.g.,:
order by
group by
sort merge join
remove duplicates
sort positions when unordered to avoid random access
Notes to remember

indexing helps navigate data faster than scan
indexing is (some times) just another way to organize data
we need to consider all levels of memory hierarchy
when we design our algorithms
and to optimally use all available bytes
Self-organizing tuple reconstruction in column-stores
Stratos Idreos, Martin Kersten, Stefan Manegold
indexing & sorting
DATA SYSTEMS
prof. Stratos Idreos