class 8

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
Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD Edgar F. Codd Innovations award

Pluto
2 years

New York
1.5 hours

this building
10 min

this room
1 min

my head
~0
random access & page-based access

need to only read $x$... but have to read all of page 1

data value $x$

page1  page2  page3  ...

data move

CPU
registers
on chip cache
on board cache
memory
disk
it all starts with how we layout the data (bits)

row-store and column-store are just two extremes in the design space
as you are starting your projects, remember

come to office hours/lab - read/post in piazza

distributed API, DSL, code is supposed to help you start fast
diverging is perfectly OK

**functionality goal:**

```sql
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:**

scalability (cores/queries)
cache conscious
vectorwised processing: how to

select max(A) from R where B<20

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

rewrite to

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

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

take plans from here:

Extra: 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
midterms

how to prepare

- open book, notes, no laptop/discussion
- material from lectures, “browse/read” readings
- 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

Saturday & Sunday before midterm: Office hours 10am-3/5pm with each one of the five TFs and Stratos (noon-1pm)
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
values are out of order

sort or even better cluster at page boundaries

intermediate out of order
initial state
columns in
insertion order

sorted A B C
propagate
order of A
\textbf{select} \texttt{max(D),min(E) from R where (A>10 and A<40) and (B>20 and B<60)}

\textbf{avoid scan} of A
\textbf{avoid TR on B}
\textbf{work on a restricted area}
\textbf{across all columns}
good for memory hierarchy

\begin{center}
\begin{tikzpicture}
  \node (A) at (0,0) {A};
  \node (B) at (2,0) {B};
  \node (D) at (4,0) {D};
  \node (pos1) at (1,-1) {pos1};
  \node (pos2) at (1,-2) {pos2};
  \node (maxD) at (5.5,0) {maxD};
  \node (1010) at (3,-1) {1010};

  \draw[->] (A) -- (B);
  \draw[->] (B) -- (D);
  \draw[->] (pos1) -- (pos2);
  \draw[->] (pos2) -- (1010);
  \draw[->] (1010) -- (maxD);

  \node at (0,-3) {sorted};
  \node at (2,-3) {for all B values between pos1 & 2: if B>20 and B<60 mark bit vector at pos i};
  \node at (4,-3) {for each marked position max(D)};
\end{tikzpicture}
\end{center}
\textbf{select} \ \text{max}(D), \text{min}(E) \ \textbf{from} \ R \ \textbf{where} \ (A>10 \ \text{and} \ A<40) \ \textbf{or} \ (B>20 \ \text{and} \ B<60)
queries that filter on A benefit

queries that filter on B benefit

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
initial state columns in insertion order

```
A
B
C
```

base data

storage budget $<\ll$ smaller than the possible set of projections

Browse: **Self-organizing tuple reconstruction in column-stores**
Stratos Idreos, Martin Kersten, Stefan Manegold
In Proc. of the ACM *SIGMOD* Int. Conference on Management of Data, 2009
cost to sort array $C_s$?
cost to find a value once sorted $C_a$?
optimized algorithm to minimize $C_s$ & $C_a$

data does not fit in L1 memory; it fits in L2
CPU can read/write directly from/to L1 only

(assume simplified memory hierarchy)
memory level L

memory level L+1

initial state: 8 unordered pages

(size=3 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)

(size=3 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)

(size=3 pages)
initial state: 8 unordered 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 sorted pages
memory level L

merge to new page

memory level L+1

(size=3 pages)

initial state: 8 sorted pages
merge to new page

memory level L

memory level L+1

(initial state: 8 sorted pages)

(size=3 pages)
merge to new page

(memory level L)

(initial state: 8 sorted pages)

(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)

memory level L

memory level L+1

(initial state: 8 sorted 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)
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 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)

\[ \log_2(N) + 1 \]
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)

2N(\log_2(N)+1) \times \text{bytesPerPage}
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 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/cluster ids/positions 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
Read textbook: Chapter 13

Browse: **Self-organizing tuple reconstruction in column-stores**
Stratos Idreos, Martin Kersten, Stefan Manegold
indexing & sorting
DATA SYSTEMS
prof. Stratos Idreos