class 7

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
research tuesday coming up
9/29 6:30pm Pierce 213 + blackboard
(chocolate included)

can we scan big data without even touching the data?
can we do joins without doing joins?

wilson, class 15
alex, class 16
& stratos

no handouts + recording
Laura Haas

Data Systems Researcher

Director of IBM Research’s Accelerated Discovery Lab & Harvard alumna

October 5

a 1 hour discussion with students will follow after class

1 hour wics meeting 3:30-4:30

The Power Behind the Throne: Information Integration in the Age of Data-Driven Discovery
Balancing Recency and Continuity in Massive Scale Dynamic Interaction Graphs

October 5, 3:30pm
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:**

```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

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

rewrite 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
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

Pluto
2 years

New York
1.5 hours

this building
10 min

this room
1 min

my head
~0

100Kx disk

100x memory

10x on board cache

2x on chip cache

registers
random access & page-based access

need to only read $x$... but have to read all of page 1
today+3
data access made better
it all starts with the select operator
filtering data: point/range queries

index knows order about 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 A B C

select max(B) from R where A<10
values are out of order
values are out of order

A
a1
a2
a3
a4
a5
B
b1
b2
b3
b4
b5
C
c1
c2
c3
c4
c5

A
5
3
2
1
4
B
a5
a3
a2
a1
a4
C
b1
b2
b3
b4
b5
C
c1
c2
c3
c4
c5
values are out of order

a query that select on A and then needs B
intermediate out of order
access should be better than scan

need more storage
tuple reconstruction will be slower
updates will be slower
+ we have to sort data in the first place
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 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

(initial state: 8 unordered pages)

memory level L-1

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

memory level L

(size=3 pages)

memory level L-1

initial state: 8 unordered pages
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

(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 sorted pages
merge to new page

memory level $L$

memory level $L-1$

(initial state: 8 sorted pages)
initial state: 8 sorted pages

merge to new page

memory level L

memory level L-1

(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

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

memory level L

(initial state: 8 sorted pages)

memory level L-1

(size=3 pages)
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)
1 pass to sort each page (2N pages)
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)
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 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)

1 pass to sort each page (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

&

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?

$$\log_{M-1}(N/M) + 1 \leq p$$
initial state
columns in
insertion order

sorted A  B  C
initial state
columns in
insertion order

sorted A  B  C

propagate
order of A

sorted A  B  C
\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
\textbf{good for memory hierarchy}

\begin{itemize}
  \item binary search for 10 & 40
  \item for all B values between pos1 & 2: if B > 20 \text{ and } B < 60
  \item mark bit vector at pos i
  \item for each marked position
  \item max(D)
\end{itemize}
\textbf{select max(D), min(E) from R where (A>10 and A<40) or (B>20 and B<60)}

\begin{itemize}
\item \textbf{binary search for 10 & 40}
\item \textbf{sorted}
\item \textbf{for all B values outside pos1 & 2: if B>20 and B<60 mark bit vector at pos i}
\item \textbf{for each marked position max(D)}
\end{itemize}
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

A    B   C

sorted
B    A   C

…

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

indexes/views/tuning knobs

but ... db cracking, adaptive* ideas

db system
other usage of sorting, e.g.,:
order by
group by
sort merge join
remove duplicates
sort positions when unordered to avoid random access
textbook: Chapter 13

Self-organizing tuple reconstruction in column-stores
Stratos Idreos, Martin Kersten, Stefan Manegold
indexing & sorting

DATA SYSTEMS

prof. Stratos Idreos

HARVARD
School of Engineering and Applied Sciences