class 6

column stores 3.0

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
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
disk

A  B  C  D

memory

option 1

A

column-store engine

option 2

A  B  C

early tuple reconstruction/materialization

row-store engine
working over fixed width & dense columns

**select**

```c
for (i=0; i<size; i++)
    if column[i] > v
        res[j++] = i
```

no function calls, no indirections, no auxiliary data, min ifs
easy to prefetch next data values

**fetch**

```c
for (i=0; i<size; i++)
    inter2[j++] = column[inter1[i]]
```
late tuple reconstruction/materialization
only reconstruct to present results

no need to assemble tuples
minimize memory footprint
minimize data we are moving up the memory hierarchy
but requires new processing engine
```
select min(C) from R where A<10 & B<20
```
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5 < R.a < 20 and 40 < R.b < 50 and 30 < S.a < 40

1. inter1 = \text{select}(Ra,5,20)
2. inter2 = \text{reconstruct}(Rb,\text{inter1})
3. inter3 = \text{select}(\text{inter2},40,50)
4. \text{join}_\text{input}_R = \text{reconstruct}(Rc,\text{inter3})
5. inter4 = \text{select}(Sa,50,65)
6. inter5 = \text{reconstruct}(Sb,\text{inter4})
7. \text{join}_\text{input}_S = \text{reverse}(\text{inter5})
8. \text{join}_\text{res}_R_S = \text{join}(\text{join}_\text{input}_R,\text{join}_\text{input}_S)
9. inter6 = \text{voidTail}(\text{join}_\text{res}_R_S)
10. inter7 = \text{reconstruct}(Ra,\text{inter6})
11. result = \text{sum}(\text{inter7})
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Ra</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
<tr>
<td>Rb</td>
<td>Rb</td>
</tr>
<tr>
<td>12</td>
<td>34</td>
</tr>
<tr>
<td>34</td>
<td>53</td>
</tr>
<tr>
<td>23</td>
<td>64</td>
</tr>
<tr>
<td>65</td>
<td>37</td>
</tr>
<tr>
<td>33</td>
<td>53</td>
</tr>
<tr>
<td>21</td>
<td>61</td>
</tr>
<tr>
<td>29</td>
<td>32</td>
</tr>
<tr>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>Rc</td>
<td>Rc</td>
</tr>
<tr>
<td>12</td>
<td>17</td>
</tr>
<tr>
<td>34</td>
<td>49</td>
</tr>
<tr>
<td>75</td>
<td>58</td>
</tr>
<tr>
<td>45</td>
<td>99</td>
</tr>
<tr>
<td>49</td>
<td>64</td>
</tr>
<tr>
<td>58</td>
<td>37</td>
</tr>
<tr>
<td>97</td>
<td>53</td>
</tr>
<tr>
<td>75</td>
<td>61</td>
</tr>
<tr>
<td>42</td>
<td>32</td>
</tr>
<tr>
<td>55</td>
<td>50</td>
</tr>
</tbody>
</table>

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra, 5, 20)
2. inter2 = reconstruct(Rb, inter1)
3. inter3 = select(inter2, 40, 50)
4. join_input_R = reconstruct(Rc, inter3)
5. inter4 = select(Sa, 50, 65)
6. inter5 = reconstruct(Sb, inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R, join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra, inter6)
11. result = sum(inter7)
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5 < R.a < 20 and 40 < R.b < 50 and 30 < S.a < 40

1. $\text{inter1} = \text{select}(R.a, 5, 20)$
2. $\text{inter2} = \text{reconstruct}(R.b, \text{inter1})$
3. $\text{inter3} = \text{select}(\text{inter2}, 40, 50)$
4. $\text{join_input_R} = \text{reconstruct}(R.c, \text{inter3})$
5. $\text{inter4} = \text{select}(S.a, 50, 65)$
6. $\text{inter5} = \text{reconstruct}(S.b, \text{inter4})$
7. $\text{join_input_S} = \text{reverse}(\text{inter5})$
8. $\text{join_res_R_S} = \text{join}(\text{join_input_R}, \text{join_input_S})$
9. $\text{inter6} = \text{voidTail}(\text{join_res_R_S})$
10. $\text{inter7} = \text{reconstruct}(R.a, \text{inter6})$
11. $\text{result} = \text{sum}(\text{inter7})$
**Initial Status**

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

**Query and Query Plan (MAL Algebra)**

\[
\text{select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40}
\]

1. \(\text{inter1} = \text{select}(\text{Ra},5,20)\)
2. \(\text{inter2} = \text{reconstruct}(\text{Rb},\text{inter1})\)
3. \(\text{inter3} = \text{select}(\text{inter2},40,50)\)
4. \(\text{join_input_R} = \text{reconstruct}(\text{Rc},\text{inter3})\)
5. \(\text{inter4} = \text{select}(\text{Sa},50,65)\)
6. \(\text{inter5} = \text{reconstruct}(\text{Sb},\text{inter4})\)
7. \(\text{join_input_S} = \text{reverse}(\text{inter5})\)
8. \(\text{join_res_R_S} = \text{join}(\text{join_input_R},\text{join_input_S})\)
9. \(\text{inter6} = \text{voidTail}(\text{join_res_R_S})\)
10. \(\text{inter7} = \text{reconstruct}(\text{Ra},\text{inter6})\)
11. \(\text{result} = \text{sum}(\text{inter7})\)

**Diagram:**

- **join(join_input_R,join_input_S)**
  - join_input_R: 4 23 9 29
  - join_input_S: 62 3 29 5 19 7 81 8 23 10
  - join_res_R_S: 4 10 9 5
- **voidTail(join_res_R_S)**
  - join_res_R_S: 4 10 9 5
  - inter6: 4 9
Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
update row7=(A=a, B=b, C=c, D=d)

A B C D

vs

A B C D

cost: 1 page

cost: N pages, N=# of columns
A | B | C
---|---|---
tuple 1 | a1 | b1 | c1
tuple 2 | a2 | b2 | c2
tuple 3 | a3 | b3 | c3
tuple 4 | a4 | b4 | c4
tuple 5 | a5 | b5 | c5
tuple 6 | a6 | b6 | c6

... inserts, deletes affect whole table
pending inserts

pending deletes

update = delete followed by insert

what information do we need to remember
Assume a column-store database with a table R(A,B,C,D,E). All attributes are integers. Our workload has two classes of queries:

1) select max(B), max(C), max(D), max(E) from R where A>v1
2) select B+C+D+E from R where A>v1

Should we use late or early tuple reconstruction plans? For each query, draw the 2 possible plans and the respective operators, explain which one is best and give the total cost.
For thesel A IDs B max IDs C max IDs D max result: 0000

For thesel A IDs B C D E max(B), max(C) max(D), max(E) result: □
sel A

IDs B max IDs C max IDs D max

result

late TR

sel A

IDs B C D E

max(B), max(C)
max(D), max(E)

result

hybrid
Sel A

IDs B max

IDs C max

IDs D max

Result

Late TR

Sel A

IDs B C D E

Max(B), Max(C)

Max(D), Max(E)

Result

Hybrid
sel A  IDs B max  IDs C max  IDs D max  

result 0000

late TR

max(B), max(C) max(D), max(E) result

hybrid
sel A → IDs B → max C → IDs D → max E → result

late TR

hybrid
sel A IDs B max IDs C max IDs D max

result 000

late TR

max(B), max(C) max(D), max(E)

result

hybrid
sel A IDs B max IDs C max IDs D max

result 0000

late TR

sel A IDs B C D E max(B), max(C) max(D), max(E)

result

hybrid
sel $A$ $\rightarrow$ IDs $B$ $\rightarrow$ max $C$ $\rightarrow$ max $D$ $\rightarrow$ max $E$ $\rightarrow$ result

late TR

hybrid

max($B$), max($C$) max($D$), max($E$)
sel A  IDs B  max  IDs C  max  IDs D  max

result

late TR

sel A  IDs B  C  D  E

max(B), max(C)
max(D), max(E)

result

hybrid
Sel A

IDs B max

IDs C max

IDs D max

Result

Late TR

Sel A

IDs B C D E

Max(B), max(C)

Max(D), max(E)

Result

Hybrid
sel A \rightarrow IDs B \rightarrow max \rightarrow IDs C \rightarrow max \rightarrow IDs D \rightarrow max

result

late TR

sel A \rightarrow IDs B \rightarrow C \rightarrow D \rightarrow E

max(B), max(C), max(D), max(E)

result

hybrid
The diagram illustrates a data processing workflow with several stages:

1. **sel A**: Selection of data
2. **IDs B**: Identification of data IDs
3. **max C**: Max operation on IDs B
4. **IDs D**: Identification of another set of data IDs
5. **max E**: Max operation on IDs D

The results of these operations are combined to form the final result:

- **max(B), max(C)**
- **max(D), max(E)**

The final result is a collection of data elements, as indicated by the output at the bottom of the diagram.
late TR

hybrid
late TR

hybrid
late TR

hybrid
A hybrid

late TR

A

B

C

D

E

+(B,C,D,E)

result

+(r1,r2) r3

IDs B

r1

IDs C

r2

IDs D

r4

+(r3,r4) res

IDs A

sA

r1

r2

r3

r4

compilation
IDS A $\rightarrow$ IDS B $\rightarrow$ IDS C $\rightarrow$ $(r_1, r_2)$ $\rightarrow$ IDS D $\rightarrow$ $(r_3, r_4)$ $\rightarrow$ late TR

IDS A $\rightarrow$ IDS B $\rightarrow$ IDS C $\rightarrow$ IDS D $\rightarrow$ IDS E $\rightarrow$ $(B, C, D, E)$ $\rightarrow$ result

hybrid
The image shows a diagram of a computational process with labeled steps and results. The top part of the diagram is labeled "late TR" and includes the following steps:

1. IDs of A receive a result from r1.
2. IDs of C receive a result from r2.
3. The results of r1 and r2 are combined to produce r3.
4. The results of r3 and r4 are combined to produce the final result.

The bottom part of the diagram is labeled "hybrid" and includes the following steps:

1. IDs of A receive a result from B, C, D, and E.
2. The results of B, C, D, and E are combined to produce the final result.

The diagram illustrates the flow of data and the combination of results in a computational process.
late TR

hybrid
default
late tuple reconstruction (rather safe choice)

open topic for optimization
when and how to do selective early reconstruction

issues to consider
transformation overhead
materialization overhead
extra passes over the data
it may be almost for free (sometimes in hash joins)
dynamic code generation to fit data layouts

DSM vs. NSM: CPU performance tradeoffs in block-oriented query processing
Marcin Zukowski, Niels Nes, Peter A. Boncz
International Workshop on Data Management on New Hardware (DaMoN) 2008
Compression = data & computation

- CPU
- Registers
- On-chip cache
- On-board cache
- Memory
- Disk

Compute

Data

Speed

Time

CPU

Mem

But

Data ↓

&

Computation ↑
which one gives better compression
can we do something like huffman coding
any side-effects

(check: Business Analytics in (a) Blink from readings)
fixed width is key…
ok and how do we store variable length data?

fixed width codes that point to dictionary entries
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
A disk containing columns A, B, C, and D is moved to memory. Two options are shown:

1. Option 1: Early tuple reconstruction/materialization directly from the disk.
2. Option 2: Data is first loaded into memory, then the column store engine is used.

This diagram illustrates the process of moving data from disk to memory and the different strategies available for handling the data.
but why now...
weren’t all those design options obvious in the past as well?

- moving data from disk
- moving data from memory
- computation

1) big memories
2) cpu vs memory speed
main-memory systems
optimized for the memory wall

with or without persistent data
other system categories

noSQL, new SQL, key-value stores, matlab, etc..

column-stores = bad name  modern systems
other data models

rdf, jason, xml, arrays, sciences ?
H2O: A Hands-free Adaptive Store
Ioannis Alagiannis, Stratos Idreos, and Anastassia Ailamaki
first part done: basic concepts in modern systems

coming up: indexing and fast scans
reading

The Design and Implementation of Modern Column-store Database Systems (Sections: all -4.6 & 4.8) by D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden

IEEE Data Engineering Bulletin, 35(1), March 2012
Special Issue on Column-stores (9 short overview papers)
research papers

Column-stores vs. row-stores: how different are they really?
D. Abadi, S. Madden, and N. Hachem

Positional update handling in column stores
Sándor Héman, Marcin Zukowski, Niels J. Nes, Lefteris Sidirourgos, Peter A. Boncz
In Proc. of the ACM SIGMOD Inter. Conference on Management of Data, 2010

Updating a cracked database
Stratos Idreos, Martin Kersten, Stefan Manegold

Integrating compression and execution in column-oriented database systems
Daniel J. Abadi, Samuel Madden, Miguel Ferreira
column-stores 3.0
DATA SYSTEMS
prof. Stratos Idreos