Access Path Selection in Main-Memory Optimized Data Systems

Should I Scan or Should I Probe?

Manos Athanassoulis
Harvard University

Talk at CS265, February 16th, 2018
Access Path Selection

```
SELECT x
FROM table_A
WHERE y < 10;
```

...the system chooses how to retrieve it.
Access Path Choices

- Full Base Data Scan
- Secondary Index Scan (Auxiliary copy of the data + structure)
Access Path Choices

- Full Base Data Scan
- Secondary Index Scan (Auxiliary copy of the data + structure)
Selectivity

Scan is best

Index is best

0%

why ask this question anew?
new workloads

more and more
concurrent similar
read queries

new architectures

columnar
data organization

new hardware

main-memory optimized
data systems
modern data systems for analytics
SIMD Processing

Multi-Core

Vectorized

\[ \begin{align*}
C_1 \\
C_2 \\
C_3 \\
C_4
\end{align*} \]
1:1 Scan to Result

Shared Scans
$Q_0, Q_1, Q_2, \ldots Q_n$
Are indexes ever needed?

\[ Q_0, Q_1, Q_2, \ldots Q_n \]
Are indexes ever needed?
If so, how should the optimizer choose an access path?
let us model

$$APS\ \text{ratio} = \frac{Index\ \text{Cost}}{Scan\ \text{Cost}}$$

> 1 scan

< 1 index
access path selection modeling

scan data

ScanData = \frac{N \cdot ts}{BW}

ResultWrite = sel \cdot \frac{N \cdot ts}{BW}

\begin{align*}
N & \quad \text{rows} \\
ts & \quad \text{tuple size} \\
BW & \quad \text{memory bandwidth} \\
sel & \quad \text{query selectivity}
\end{align*}

what if we have q queries?
access path selection **modeling**

**q concurrent** queries

\[
\text{ScanData} = \max \left( \frac{N \cdot ts}{BW}, N \cdot p \cdot q \right)
\]

CPU cost per query (may be too high!)

**result write**

\[
\text{ResultWrite} = \sum_{i=1}^{q} \frac{N \cdot ts}{BW} = S_q \cdot \frac{N \cdot ts}{BW}, \quad \text{s.t.} \quad S_q = \sum_{i=1}^{q} sel_i
\]

total query selectivity (sum)
ScanCost = \max\left(\frac{N \cdot ts}{BW}, N \cdot p \cdot q\right) + S_q \cdot \frac{N \cdot ts}{BW}

ScanData + PredicateEval + ResultWrite
access path selection \textit{modeling}

TreeTraversal = \log_b(N) \cdot C_M

LeavesTraversal = \text{sel} \cdot \frac{N}{b} \cdot C_M

DataTraversal = \text{sel} \cdot \frac{N \cdot \text{its}}{BW}

ResultWrite = \text{sel} \cdot \frac{N \cdot ts}{BW}

\begin{itemize}
\item $N$ \quad \text{rows}
\item $ts$ \quad \text{tuple size}
\item $BW$ \quad \text{memory bandwidth}
\item $sel$ \quad \text{query selectivity}
\item $q$ \quad \text{queries}
\item $p$ \quad \text{predicate evaluation cost (CPU)}
\item $b$ \quad \text{branching factor (fanout)}
\item $C_M$ \quad \text{cache miss latency}
\item $its$ \quad \text{index tuple size}
\end{itemize}
access path selection \textit{modeling}

\[ \text{TreeTraversal} = \log_b(N) \cdot C_M \]

\[ \text{Sort} = sel \cdot N \cdot \log_2(sel \cdot N) \cdot \frac{its}{BW} \]

\[ \text{LeavesTraversal} = \frac{N}{b} \cdot C_M \]

\[ \text{DataTraversal} = sel \cdot \frac{N \cdot its}{BW} \]

\[ \text{ResultWrite} = sel \cdot \frac{N \cdot ts}{BW} \]
$$\text{IndexCost} = q \cdot \log_b(N) \cdot C_M + S_q \cdot \left( \frac{N}{b} \cdot C_M + \frac{N \cdot \text{its}}{BW} + \frac{N \cdot \text{ts}}{BW} \right)$$

TreeTraversal + LeavesTraversal + ResultWrite

DataTraversal
new access path selection

Dynamic Parameter

$q$: #concurrent read queries
$S_q$: sum of query selectivity of $q$ queries $S_q = \sum_{i=1}^{q} sel_i$
new access path selection

\[
APS \text{ ratio} = \frac{q \cdot \log_b(N) \cdot C_M + S_q \cdot \left(\frac{N}{b} \cdot C_M + \frac{N \cdot its}{BW} + \frac{N \cdot ts}{BW}\right)}{\max \left(\frac{N \cdot ts}{BW}, q \cdot p \cdot N\right) + S_q \cdot \frac{N \cdot ts}{BW}}
\]

Dynamic Parameter

- \(q\) #concurrent read queries
- \(S_q\) sum of query selectivity of \(q\) queries \(S_q = \sum_{i=1}^{q} sel_i\)

- \(N\) rows
- \(ts\) tuple size
- \(BW\) memory bandwidth
- \(sel\) query selectivity
- \(q\) queries
- \(p\) predicate evaluation cost (CPU)
- \(b\) branching factor (fanout)
- \(C_M\) cache miss latency
- \(its\) index tuple size
new access path selection

\[ \text{APS ratio} = \frac{\log_B \left( \frac{N}{q} \right) + BW \cdot C_M + S_q \cdot \left( BW \cdot C_M + (ts + its) \right)}{\max(ts, q \cdot p \cdot BW) + S_q \cdot ts} \]

Dynamic Parameter

- \( q \)  
  \#concurrent read queries
- \( S_q \)  
  sum of query selectivity of \( q \) queries \( S_q = \sum_{i=1}^{q} sel_i \)
new access path selection

Modeling

data size has complex effect

concurrency matters!!
**new access path selection**

**Experiments**

- **8 Queries | 8 Cores**
  - **Scan**
  - **Index**
  - Selectivity Crossover vs Relation Size

- **100M Tuples | 8 Cores**
  - **Scan**
  - **Index**
  - Selectivity Crossover vs Concurrent read queries

*shared scans help up to a point*
new access path selection

Experiments

8 Queries | 8 Cores

Scan

Index

Relation Size

10^4 10^5 10^6 10^7 10^8 10^9

Selectivity Crossover

0.0% 0.2% 0.4% 0.6% 0.8% 1.0% 1.2% 1.4% 1.6% 1.8%

0.0% 0.2% 0.4% 0.6% 0.8% 1.0%

100M Tuples | 8 Cores

Scan

Index

shared scans help up to a point

Concurrent read queries

10 100
new access path selection

Experiments

Scan
Index

Selectivity Crossover

Relation Size

Selectivity Crossover

Concurrent read queries

8 Queries | 8 Cores

100M Tuples | 8 Cores

Two x 256

Index

Scan
new access path selection

Selectivity

Scan is best

0%

Index is best

Concurrent Queries

Scan is best

Selectivity

0%

Index is best
Hardware Improvements

Dawn of time 2000

10%
1%
0%
Dawn of time

2000

2010

Hardware Improvements

Column Stores

0%

1%

10%
Main Memory

Dawn of time 2000 2010 2017

Hardware Improvements

Column Stores

Main Memory
“What-if” questions