Smooth Scan: Statistics-Oblivious Access Paths

Renata Borovica-Gajic  Stratos Idreos  Anastasia Ailamaki
Marcin Zukowski  Campbell Fraser

Inspired by Dagstuhl 2012 seminar on Robust Query Processing
Optimizers’ sensitivity to statistics

Setting: TPC-H, SF10, DBMS-X, Tuning tool 5GB space

Degradation due to sub-optimal access paths
Cause for sub-optimal plans

CARDINALITY ERRORS

- Order of magnitude more tuples
- 75x longer execution time!
Access path selection problem

Re-optimization: risky

[Mid’98, Pop’04, Rio’05, Bou’14]

Statistics: unreliable advisor
Re-optimization: risky
Access paths under looking glass

Index Access
+ read what you need
- random (& repeated) I/O

Sequential Access
+ (fast) sequential I/O
- read everything

No single path is optimal
Quest for robust access paths

Near-optimal throughout entire selectivity range
Smooth Scan in a nutshell

- Statistics-oblivious access path
- Learn result distribution at run-time
- Adapt as you go

DESIGN GOALS

- Avoid performance cliffs & risk
- Continuous, gradual and smooth adaptation
Adaptivity with Smooth Scan

Morph between Index and Sequential Scan based on observed result distribution
Morphing mechanism

• Modes:
  1. **Index Access**: Traditional index access
  2. **Entire Page Probe**: Index access probes entire page
  3. **Gradual Flattening Access**: Probe adjacent region(s)
Morphing policies

- Policies:
  - Greedy
  - Selectivity Increase Driven
  - Elastic

Region snooping = Selectivity driven adaptation

Selectivity increase -> Mode Increase
$SEL_{region} > SEL_{global}$

Selectivity decrease -> Mode Decrease
$SEL_{region} < SEL_{global}$
## Smooth Scan benefits

<table>
<thead>
<tr>
<th></th>
<th>Index Scan</th>
<th>Full Scan</th>
<th>Sort Scan</th>
<th>Smooth Scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avoid repeated accesses</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fast sequential I/O</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Avoid full table read</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Tuples pipelining</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Experimental setup

Hardware:

2 Intel Xeon 6-core CPU @2.8 GHz, 48GB RAM
HDD: I/O transfer rate 120 MB/s, Random vs. Sequential ratio = 10

Software:

PostgreSQL 9.2.1: Index Scan, Full Scan, Sort (Bitmap) Scan, Smooth Scan

Workload:

TPC-H: SF 10
Micro-benchmark: 400M tuples, 10 columns random (1 – 10^5), 25GB
Q1: select * from relation where c2 >= 0 and c2< X% [order by c2];

Experimental Condition:

Cold file system cache
TPC-H with Smooth Scan

Setting: TPC-H, SF10, PostgreSQL with Smooth Scan

Robust execution for all queries
Smooth Scan significantly decreases I/O wait time

Smooth Scan significantly decreases I/O wait time
**Snooping I/O access**

*Setting*: TPC-H, Q1, Lineitem table, iosnoop tool

Smooth Scan reduces random I/O requests
Adaptivity over selectivity range

**Setting:** Micro-benchmark, Q1 (w. and w/o. order), Selectivity 0-100%

Near-optimal performance throughout entire range
Evaluation of Smooth Scan Modes

**Setting:** 400M tuples, 10 int. attributes, 25GB, Index(c2), cold runs

**Query:** `select * from R where c2 < X%;`

Full morphing prevents degradation of 115x compared to Index Scan and is 20% slower than Full Scan.
**Smooth Scan on SSD**

**SSD:** Deneva 2C Series, transfer 550MB/s, 80kI/O/s of random reads

**Query:** select * from R where c2 < X%;

- **Full Scan**
- **Index Scan**
- **Sort Scan**
- **Smooth Scan**

**Higher benefit for Smooth Scan on SSD than on HDD**
Setting: TPC-H, SF10, Warm caches

CPU overhead cannot be fully masked
Conclusions

SMOOTH SCAN

• **Statistics-oblivious** access path
• Uses region snooping to **morph** between alternatives
• **Near-optimal performance** for all selectivities

IMPACT

• Removes access path selection decision
• Robust execution for all query inputs
Open questions

• In-memory implementation
• Handling concurrency
• Index intersection
• Column stores
• Sub-optimal join order
• Something else?

Thank you!
Thank you!

renata.borovica@epfl.ch  stratos@seas.harvard.edu  natassa@epfl.ch  marcin@snowflake.net  campbellf@google.com
Bibliography