Smooth Scan: Statistics-Oblivious Access Paths

Renata Borovica-Gajic  Stratos Idreos  Anastasia Ailamaki
Marcin Zukowski  Campbell Fraser
Optimizers’ sensitivity to statistics

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

Degradation due to sub-optimal access paths
Access path selection problem

Re-optimization: risky

Statistics: unreliable advisor
Re-optimization: risky
No single path is optimal

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

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

Selectivity increase -> Mode Increase
\[ SEL_{region} > SEL_{global} \]
Selectivity decrease -> Mode Decrease
\[ SEL_{region} < SEL_{global} \]

Region snooping = Selectivity driven adaptation
# 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
## TPC-H breakdown

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q4</th>
<th>Q6</th>
<th>Q7</th>
<th>Q14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pSQL</td>
<td>Smooth S.</td>
<td>pSQL</td>
<td>Smooth S.</td>
<td>pSQL</td>
</tr>
<tr>
<td># I/O Requests (K)</td>
<td>70</td>
<td>77</td>
<td>224</td>
<td>235</td>
<td>566</td>
</tr>
</tbody>
</table>

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
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
Q & A

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

Thank you!