SMOKE: Fine-Grained Provenance Capture at Interactive Speed

Fotis Psallidas
fotis@cs.columbia.edu

Eugene Wu
ewu@cs.columbia.edu
O = \gamma_{\text{state}, \text{avg}(\text{delay})}(\text{Airports} \bowtie \text{Flights})

name = from

<table>
<thead>
<tr>
<th>Airports</th>
<th>name</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>LGA</td>
<td>NY</td>
</tr>
<tr>
<td>a₂</td>
<td>JFK</td>
<td>NY</td>
</tr>
<tr>
<td>a₃</td>
<td>MIA</td>
<td>FL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flights</th>
<th>from</th>
<th>delay</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>j₁</td>
<td>LGA</td>
<td>30</td>
<td>NY</td>
</tr>
<tr>
<td>j₂</td>
<td>LGA</td>
<td>40</td>
<td>NY</td>
</tr>
<tr>
<td>j₃</td>
<td>JFK</td>
<td>50</td>
<td>NY</td>
</tr>
<tr>
<td>j₄</td>
<td>MIA</td>
<td>60</td>
<td>FL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>state</th>
<th>avg(delay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>o₁</td>
<td>NY</td>
</tr>
<tr>
<td>o₂</td>
<td>FL</td>
</tr>
</tbody>
</table>
Fine-Grained Provenance Primer

Airports

<table>
<thead>
<tr>
<th>name</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>LGA</td>
</tr>
<tr>
<td>a₂</td>
<td>JFK</td>
</tr>
<tr>
<td>a₃</td>
<td>MIA</td>
</tr>
</tbody>
</table>

Flights

<table>
<thead>
<tr>
<th>from</th>
<th>delay</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>f₁</td>
<td>30</td>
<td>NY</td>
</tr>
<tr>
<td>f₂</td>
<td>40</td>
<td>NY</td>
</tr>
<tr>
<td>f₃</td>
<td>50</td>
<td>NY</td>
</tr>
<tr>
<td>f₄</td>
<td>60</td>
<td>FL</td>
</tr>
</tbody>
</table>

\{f₄\} = \text{backward\_trace}(\{o₂\}, \text{Flights})
\{o_2\} = \text{forward_trace}\{\{f_4\}, \mathcal{O}\}
Fine-Grained Provenance Primer

Airports
- LGA: NY
- JFK: NY
- MIA: FL

Flights
- $f_1$: LGA 30
- $f_2$: LGA 40
- $f_3$: JFK 50
- $f_4$: MIA 60

Delay and State
- $o_1$: NY 40
- $o_2$: FL 60
Navigation of the input-output connections
  - \{\text{records}\} = \text{backward_trace}(\ldots)
  - \{\text{records}\} = \text{forward_trace}(\ldots)

Provenance consuming queries
  - SQL(\text{backward_trace}(\ldots))
  - SQL(\text{forward_trace}(\ldots))
Fine-Grained Provenance Capture

Optimizing Provenance Constructs ⇒ Fine Grained Provenance Capture Problem

Fine-grained Provenance Capture Problem
Capture provenance graph w/ low-overhead to answer provenance queries efficiently
Previous Work

**LAZY** [Cui et al. and Ikeda et al.]
- Do not materialize the provenance graph
- Provenance query evaluation by rewriting to SQL queries

**EAGER** [Subzero, NewT, Ramp, Clothia et al., Titian, Trio, Perm, GProm, ProQL and DBNotes]
- Materialize the provenance graph
  - **Logical**: In the relational model
  - **Physical**: In a separate provenance subsystem
- Provenance query evaluation using the materialized graph
Main Problems

Physical
- Expensive virtual function calls
- Write-inefficient provenance storage
- No physical plan co-optimizations

Logical
- Storing graph under relational model
- Hard to optimize queries
- Expensive joins
Previous Work – Provenance Query

Main Problems

Lazy Approaches
- Equivalent SQL queries are slow

Physical Approaches
- Read-inefficient provenance storage

Logical Approaches
- Need extra indexing steps

Cum. Latency (ms, log) over 5000 backward trace queries
Design Principles for Fast Provenance Systems

4 Design Principles

P1. Tight Integration
Integrate provenance capture within query plans
Index provenance in read- and write-efficient indexes

P2. Reuse work
Provenance indexes in hash tables
Intra-plan hash table reuse

P3. Apriori Knowledge
Don’t capture if not used

P4. Provenance Consumption
Push computation into provenance capture
SMOKE Overview

- In-Memory Query Compiled Engine
- Eager physical approach
- Write and read-efficient provenance indexes
- Two capture paradigms: Defer and Inject
Provenance Index Representations

input

\[
\begin{array}{c}
R \\
r_1 \\
r_2 \\
r_3 \\
\ldots
\end{array}
\]

\[\Rightarrow\]

output

\[
\begin{array}{c}
O \\
o_1 \\
o_2 \\
o_3 \\
o_4
\end{array}
\]

rid index

\[
\begin{array}{c}
1\text{-to-}N \\
r_1 \\
r_4 \\
r_7 \\
r_5 \\
r_8 \\
r_9 \\
r_3
\end{array}
\]

rid array

\[
\begin{array}{c}
1\text{-to-}1 \\
r_2 \\
r_4 \\
r_2
\end{array}
\]
Two Capture Paradigms

Inject

Defer

<table>
<thead>
<tr>
<th>State</th>
<th>Avg (Delay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>40</td>
</tr>
<tr>
<td>FL</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From</th>
<th>Delay</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGA</td>
<td>30</td>
<td>NY</td>
</tr>
<tr>
<td>LGA</td>
<td>40</td>
<td>NY</td>
</tr>
<tr>
<td>JFK</td>
<td>50</td>
<td>NY</td>
</tr>
<tr>
<td>MIA</td>
<td>60</td>
<td>FL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Avg (Delay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>40</td>
</tr>
<tr>
<td>FL</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From</th>
<th>Delay</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGA</td>
<td>30</td>
<td>NY</td>
</tr>
<tr>
<td>LGA</td>
<td>40</td>
<td>NY</td>
</tr>
<tr>
<td>JFK</td>
<td>50</td>
<td>NY</td>
</tr>
<tr>
<td>MIA</td>
<td>60</td>
<td>FL</td>
</tr>
</tbody>
</table>
Both paradigms for every relational operator
Avoid materializing intermediate provenance indexes for multi-operator plans
Experimental Setup

- Provenance Capture and Querying
  - Comparison with state-of-the-art Physical, Logical, and Lazy approaches
  - Single operator performance: microbenchmarks using zipfians
  - Multi-operator performance: TPC-H

- Real-World Applications
  - Comparison with state-of-the-art, hand-tuned implementations
  - Interactive Visualizations:
    - Overview first then details on demand
    - Crossfilter
  - Data Profiling: Functional Dependency Checks
Main Results - Provenance Capture

Smoke outperforms alternatives by embodying the design principles

Negligible overhead over fast query engines

Can do better with opts (.8x → .4x)

GB --- 10M Tuples, 100 groups, $\theta = 1$

<table>
<thead>
<tr>
<th>Method</th>
<th>Capture Latency (ms, log)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>183x</td>
</tr>
<tr>
<td>Logical</td>
<td>16.9x</td>
</tr>
<tr>
<td>Smoke-D</td>
<td>1.4x</td>
</tr>
<tr>
<td>Smoke-I</td>
<td>0.8x</td>
</tr>
<tr>
<td>No Capture</td>
<td></td>
</tr>
</tbody>
</table>

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science
Main Results - CROSSFILTER

Provenance can express important application logic

**Example:** Show the distribution of #flights per carrier only for selected states

```
backward_trace(airports, flights)
```

```
selective_refresh(V2, flights)
```

Example: Show the distribution of #flights per carrier only for selected states

- AA
- CO
- NW
- WN
- UA
- HA
- DL
- FL
- OO

#flights
Main Results - CROSSFILTER

Crossfilter on Ontime dataset: 5 initial views over 125 million delayed flights

Data cubes: Immens > 1 hour, Nanocubes > 30 minutes, Hashedcubes > 4 minutes
<table>
<thead>
<tr>
<th>Interactive Visualizations</th>
<th>Interactive Data Profiling</th>
<th>Multi-Application Linking</th>
<th>Interactive Query Specification</th>
<th>What-if Provisioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Explanations</td>
<td>Why-not Analytics</td>
<td>Iterative Analytics</td>
<td>Viz Workflow Debugging</td>
<td>Interactive Data Cleaning</td>
</tr>
<tr>
<td>Interaction Debugging</td>
<td>ML Interpretability</td>
<td>Visualization Deconstruction and Restyling</td>
<td>Interaction By Example</td>
<td>Application Design Search</td>
</tr>
<tr>
<td>Collaborative Communication</td>
<td>Action Recovery</td>
<td>Sense-Making</td>
<td>Meta-Analysis</td>
<td>Replication and Reproducibility</td>
</tr>
<tr>
<td>Data Integration</td>
<td>Debugging</td>
<td>Network Diagnostics</td>
<td>Dataset Versioning</td>
<td>Auditing</td>
</tr>
</tbody>
</table>

...
Thank you

Smoke: Fine-Grained Lineage At Interactive Speed
A Deep breath of Data-Intensive Lineage Applications
Provenance for Interactive Visualizations
Combining Design and Performance in a DVMS

[VLDB18]
[SIGMOD18]
[HILDA18]
[CIDR17]