Ibis: A Provenance Manager for Multi-Layer Systems

Christopher Olston & Anish Das Sarma
Yahoo! Research
Motivation: Many Sub-Systems

- Scalable file system (e.g., GFS)
- Distributed sorting & hashing (e.g., Map-Reduce)
- Dataflow programming framework (e.g., Pig)
- Workflow manager (e.g., Oozie)
- Low-latency processor
- Serving
- Ingestion

Metadata queries

Datum X

Datum Y

Provenance of X?
Ibis Project

- **Benefits:**
  - Provide uniform view to users
  - Factor out metadata management code
  - Decouple metadata lifetime from data/subsystem lifetime

- **Challenges:**
  - Overhead of shipping metadata
  - Disparate data/processing granularities

THIS PAPER
Example Granularity Lattices

data granularities

- Web page
- Row
- Column
- Table
- Version
- Cell

process granularities

- Workflow
- Pig script
- Pig logical operation
- Pig physical operation
- MR job
- MR job phase
- MR task
- Task attempt
- MR program
- Pig job
Challenges

• **Inference:** Given relationships expressed at one granularity, answer queries about other granularities *(the semantics are tricky here!)*

• **Efficiency:** Implement inference without resorting to materializing everything in terms of finest granularity *(e.g. cells)*
Talk Outline

• Informal overview
  – Example data provenance graph
  – Query language overview + examples

• Touch on formal model (details in paper)
Example Workflow
Provenance Graph

IMDB extracted table

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>lead actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avatar</td>
<td>2009</td>
<td>Worthington</td>
</tr>
<tr>
<td>Inception</td>
<td>2010</td>
<td>DiCaprio</td>
</tr>
</tbody>
</table>

Yahoo extracted table

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>lead actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avatar</td>
<td>2009</td>
<td>Saldana</td>
</tr>
<tr>
<td>Inception</td>
<td>2010</td>
<td>DiCaprio</td>
</tr>
</tbody>
</table>

combined table

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>lead actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avatar</td>
<td>2009</td>
<td>V1: Worthington</td>
</tr>
<tr>
<td>Avatar</td>
<td>2009</td>
<td>V2: Saldana</td>
</tr>
<tr>
<td>Inception</td>
<td>2010</td>
<td>DiCaprio</td>
</tr>
</tbody>
</table>

pig job 1

version = 2
wrapper = imdb

pig job 2

version = 3
wrapper = yahoo

extract pig script

map task 1, attempt 1
map task 2, attempt 1
reduce task 1, attempt 1
merge pig script
Meaning of Provenance Relationships

• (P, D1, D2): Process P consumed \textit{PART OF} datum D1 and emitted \textit{ALL OF} datum D2
• “part\rightarrow all” semantics are a natural default
• Upshot: if D1 and D2 are tables, cannot infer that a given row in D1 influenced D2
• In query language, can still ask “part\rightarrow part” questions:
  \[ \exists d2 \in D2 \text{ such that } D1 \text{ influenced } d2? \]
Query Language: “IQL”

- SQL-style language for querying the provenance graph

- Special constructs:
  - **Under (containment):** Is row R under table T?
  - **Influence:** Does data D1 influence data D2?
  - **Feed:** Does data D feed process P?
  - **Emit:** Does process P emit data D?
IQL Examples

• Find data items that influenced the combined extracted table:

```sql
select d.id
from AnyData d, Table t
where d influences t and
  t.id = (combined extracted table);
```

• Find data tables that are “contaminated” by version 3 of the extraction script (found to have a bug):

```sql
select t.id
from PigScript p, PigJob j,
     AnyData d1, AnyData d2, Table t
where p.id = (extract pig script) and j under p
  and j.version = 3 and j emits d1
  and d1 influences d2 and d2 under t;
```
Implementation Status

• We have a working storage/query engine based on rewriting over SQL/RDBMS (SQLite)
• We’re currently working on automatic provenance capture (from Pig, Hadoop, etc.)
Talk Outline

• Informal overview
  – Example data provenance graph
  – Query language overview + examples

Touch on formal model (details in paper)
  – Open-world semantics
  – Transitive inference of containment & influence
Open-World Semantics

- Metadata of Ibis encodes set $F$ of facts
- Open-world:
  - Correctness: All facts in $F$ are correct
  - Incomplete: May be other facts unknown to Ibis
- Extension, $\text{ext}(F)$, of facts that can be derived from $F$
- True world has set of facts $F'$
- We have $F \subseteq \text{ext}(F) \subseteq F'$
Open-World Semantics:
One Implication

• Suppose F contains:
  • Process p emitted row r1
  • Currently r1 is the only row in table T
• ``Process p emitted table T’’ is a fact that may be in F’ (true world) but cannot be inferred in ext(F)
Inferring “X is under Y”

• Defined in terms of “granularization”:
  1. Resolve X and Y into finest-grain elements (e.g. cells)
  2. Perform set containment check

• Implemented via a shortcut that avoids enumerating sub-elements

• Proof that implementation & definition are equivalent
## Inferring “X is under Y”

<table>
<thead>
<tr>
<th>Basic element $b$ defined by granularity $g$, direct <em>parents</em> $P$ (and an identifier).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Granularization of</strong> $b$ <strong>to finest granularity</strong> $g_{\text{min}}$ <strong>defined by:</strong></td>
</tr>
<tr>
<td>$G(b) = {b'=(g_{\text{min}}, P') \mid b \text{ contains } b'}$</td>
</tr>
<tr>
<td><strong>Containment</strong> obtained by recursive application of parent relation</td>
</tr>
<tr>
<td><strong>Complex element</strong> $E$ <strong>defined by set of granularity</strong> ${g_1, \ldots, g_n}$, and corresponding basic elements ${b_1, \ldots, b_n}$.</td>
</tr>
<tr>
<td><strong>Granularization of complex element</strong> $E$ <strong>consisting of</strong> $b_1, \ldots, b_n$ <strong>is:</strong></td>
</tr>
<tr>
<td>$G(E) = \bigcap_i G(b_i)$</td>
</tr>
</tbody>
</table>
Inferring “X is under Y”

Under Check-1: Sets of complex elements $E_1, E_2$. $E_1$ is under $E_2$ iff not exists a true world with $\bigcup_{e_1 \in E_1} G(e_1) \not\subseteq \bigcup_{e_1 \in E_1} G(e_1)$

Efficient Under Check-2: Sets of complex elements $E_1, E_2$. $E_1$ is under $E_2$ iff for all $e_1 \in E_1$, exists $e_2 \in E_2$ such that $e_1$ is under $e_2$.

Given complex elements $e_1$ and $e_2$ with basic element sets $B(e_1)$ and $B(e_2)$, $e_1$ is under $e_2$ iff for all $b_2 \in B(e_2)$, exists $b_1 \in B(e_1)$ such that $b_2$ contains $b_1$.

Theorem: Check-1 is equivalent to Check-2.
Inferring “X influences Y”

Given two data vertices d1 and d2:

(1) d1 influences(0) d2 iff d2 is under d1;

(2) d1 influences(1) d2 iff one of the following hold:
   (A) d1 influences(0) d2
   (B) there exists a provenance relationship (d1’,p,d2’) such that d1 influences(0) d1’ and d2’ influences(0) d2

(3) For any integer k>1, d1 influences(k) d2 iff exists d* such that d1 influences(1) d* and d* influences(k-1) d2
Related Work

• Multi-layer system provenance:
  – Harvard PASSv2

• Nested collections in scientific workflow provenance:
  – Kepler’s COMAD nested collections
  – ZOOM user views
  – Open provenance model

• Annotations on arbitrary sub-regions of relations:
  – [Eltabakh et al.]
  – [Srivastava et al.]
Summary

• Many semi-independent data mgmt. layers + provenance query needs $\rightarrow$ integrated provenance

• Diverse data & process granularities $\rightarrow$ careful semantics

• Our contributions:
  – Formal multi-granularity provenance semantics
  – Query language
  – Working prototype (see paper; work in progress)