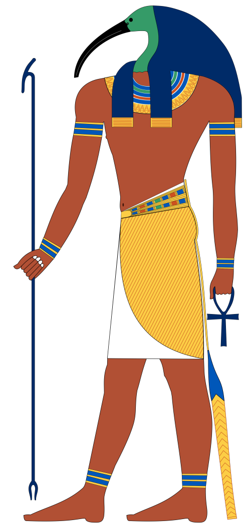
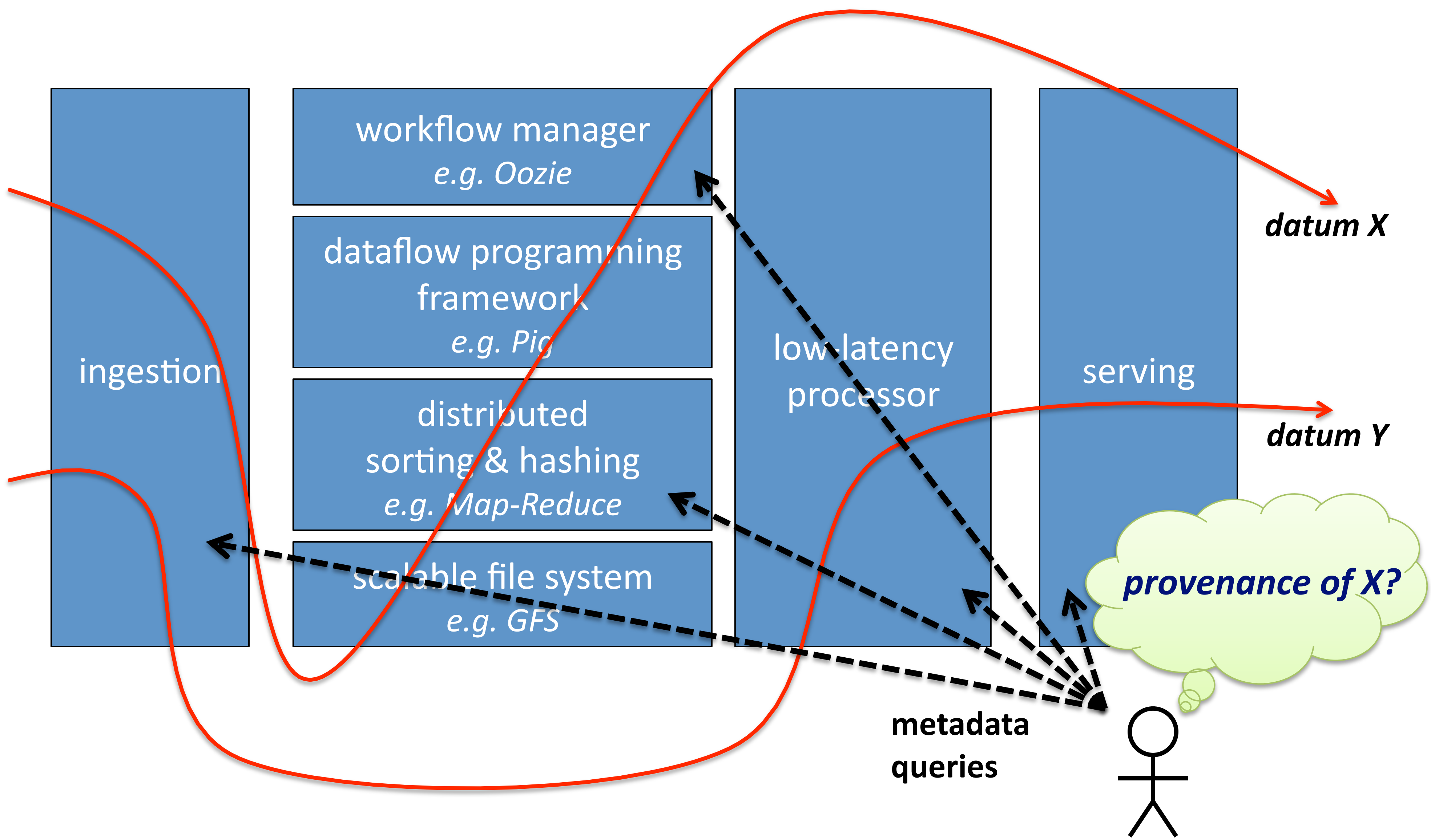
Ibis: A Provenance Manager for Multi-Layer Systems



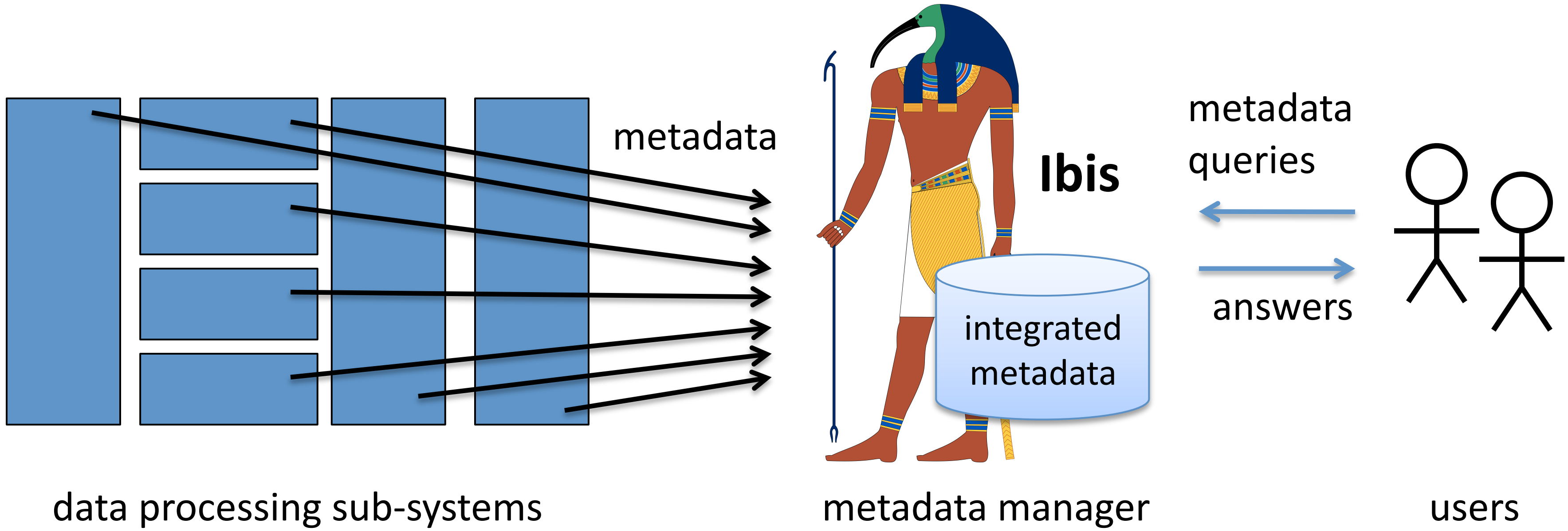
Christopher Olston & Anish Das Sarma

Yahoo! Research

Motivation: Many Sub-Systems



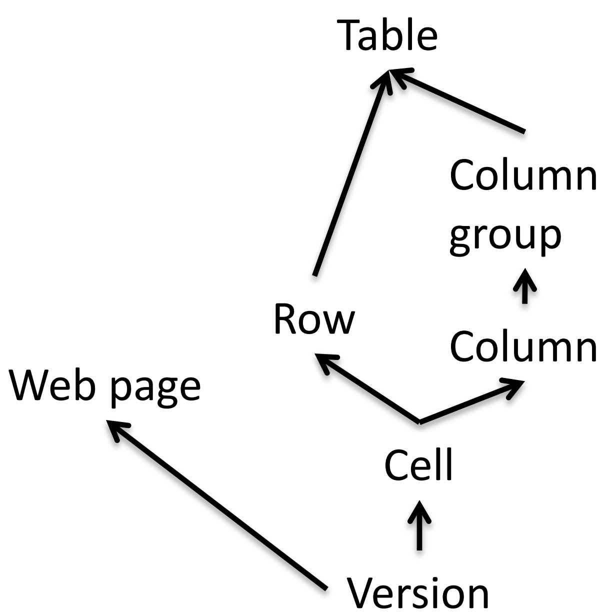
Ibis Project



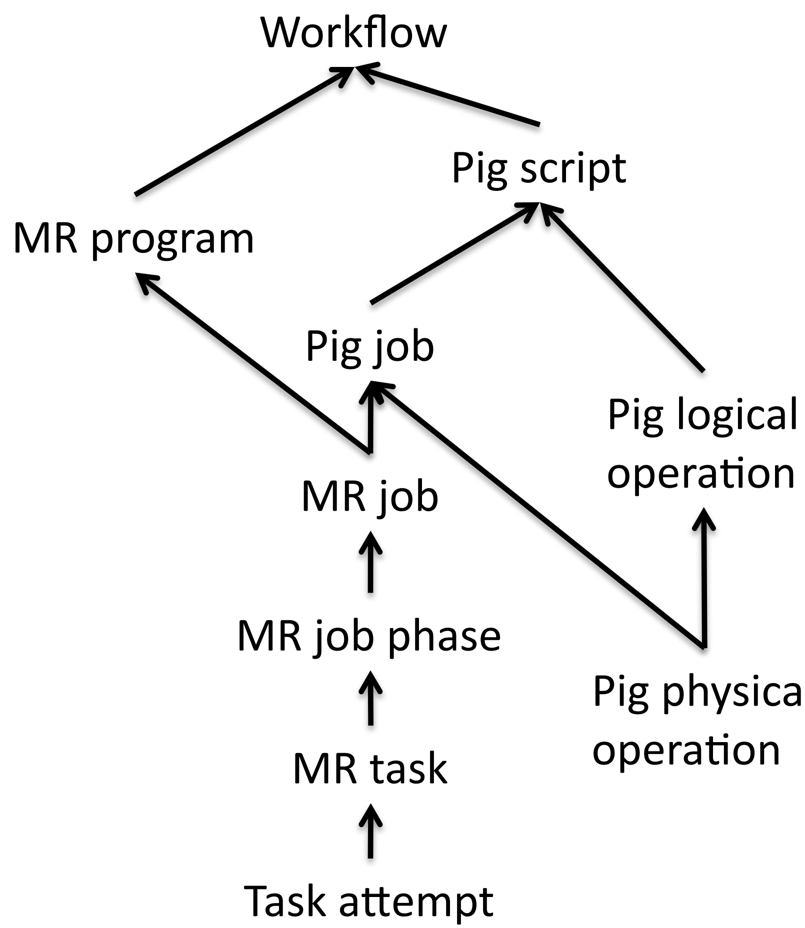
- Benefits:
  - Provide uniform view to users
  - Factor out metadata management code
  - Decouple metadata lifetime from data/subsystem lifetime

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

Example Granularity Lattices



data granularities



process granularities

Challenges

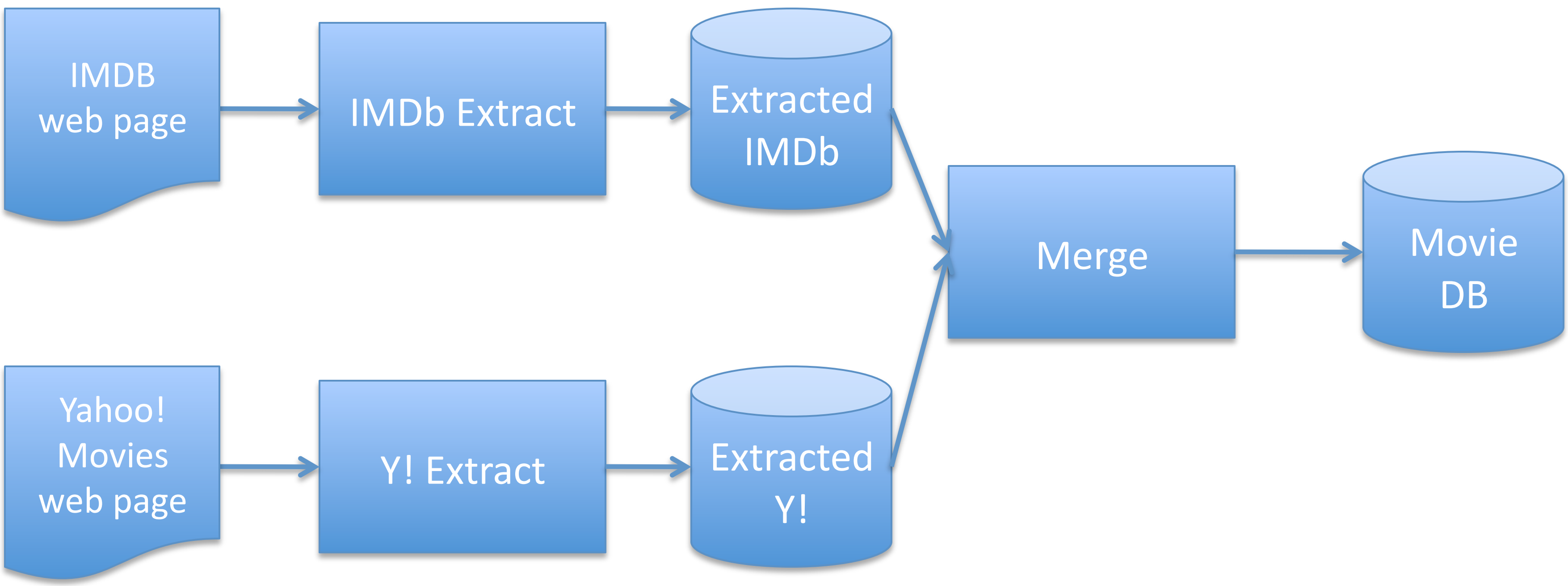
- ***Inference:*** Given relationships expressed at one granularity, answer queries about other granularities *(the semantics are tricky here!)*

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

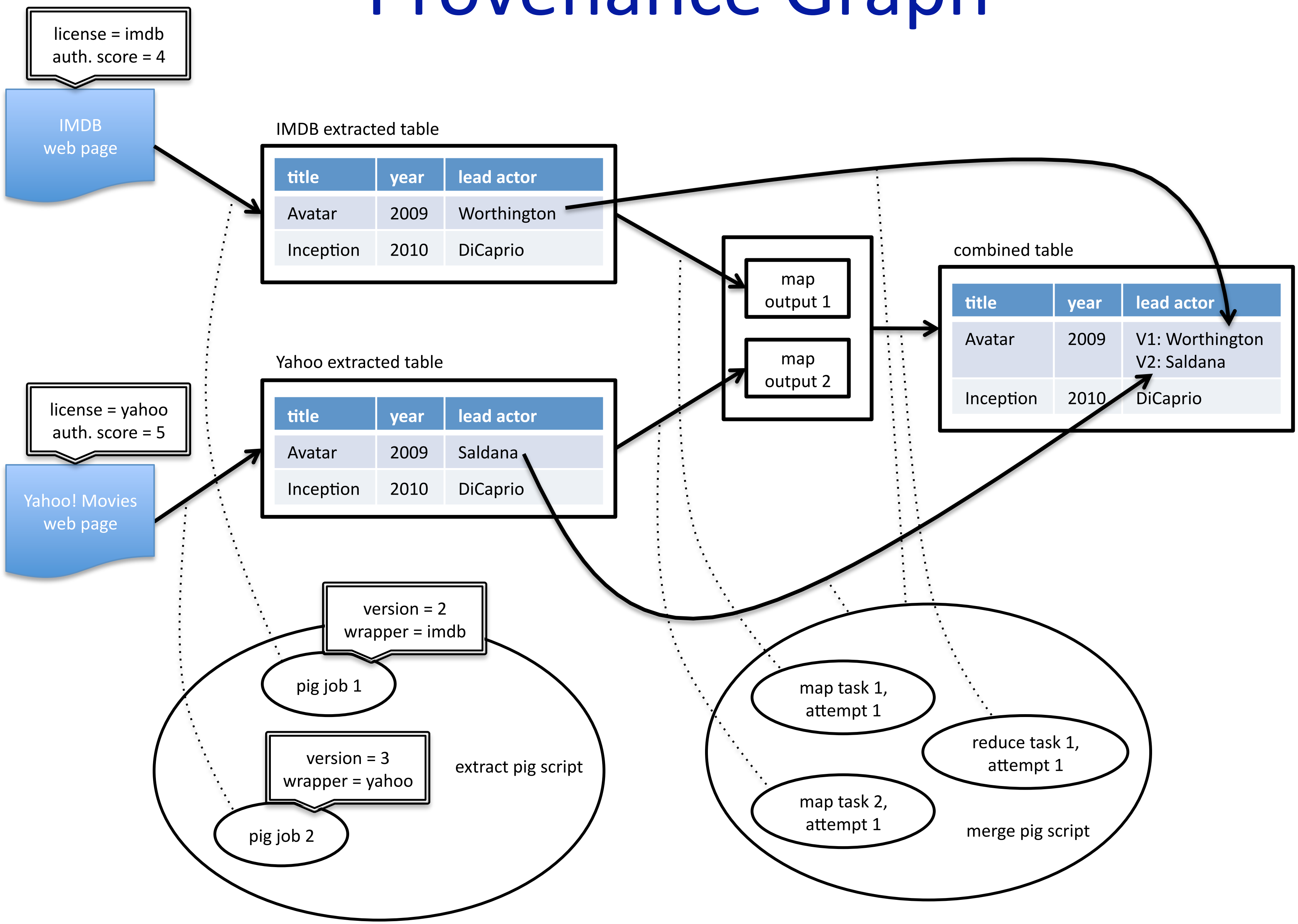
Talk Outline

- Informal overview
  - Example data provenance graph
  - Query language overview + examples
- Touch on formal model (details in paper)

Example Workflow



Provenance Graph



Meaning of Provenance Relationships

- (P, D1, D2): Process P consumed ***PART OF*** datum D1 and emitted ***ALL OF*** datum D2
- "part→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→part" questions:

  ∃d2 ε D2 such that *D1 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:

```
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):

```
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, ext(F), of facts that can be derived from F
- True world has set of facts F'
- We have $F \subseteq ext(F) \subseteq F'$

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"

Basic element b defined by granularity g, direct *parents* P (and an identifier).

Granularization of b to finest granularity g_{min} defined by:

$$G(b) = \{b' = (g_{min}, P') \mid b \text{ contains } b'\}$$

Containment obtained by recursive application of parent relation

Complex element E defined by set of granularity $\{g_1,\ldots,g_n\}$, and corresponding basic elements $\{b_1,\ldots,b_n\}$.

Granularization of complex element E consisting of $b_1, \ldots., b_n$ is:

$$G(E) = \cap_i G(b_i)$$

Inferring “X is under Y”

Under Check-1: Sets of complex elements E1, E2. E1 *is under* E2 iff not exists a true world with $\cup_{e1 \varepsilon E1} G(e1) \not\subseteq \cup_{e1 \varepsilon E1} G(e1)$

Efficient Under Check-2: Sets of complex elements E1, E2. E1 *is under* E2 iff for all $e_1 \varepsilon E_1$, exists $e_2 \varepsilon E_2$ such that *e1 is under e2.*

Given complex elements e1 and e2 with basic element sets B(e1) and B(e2), e1 is under e2 iff for all $b_2 \varepsilon B(e_2)$, exists $b_1 \varepsilon B(e_1)$ 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 → integrated provenance
- Diverse data & process granularities → careful semantics
- Our contributions:
  - Formal multi-granularity provenance semantics
  - Query language
  - Working prototype (see paper; work in progress)