Putting Lipstick on Pig: Enabling Database-style Workflow Provenance

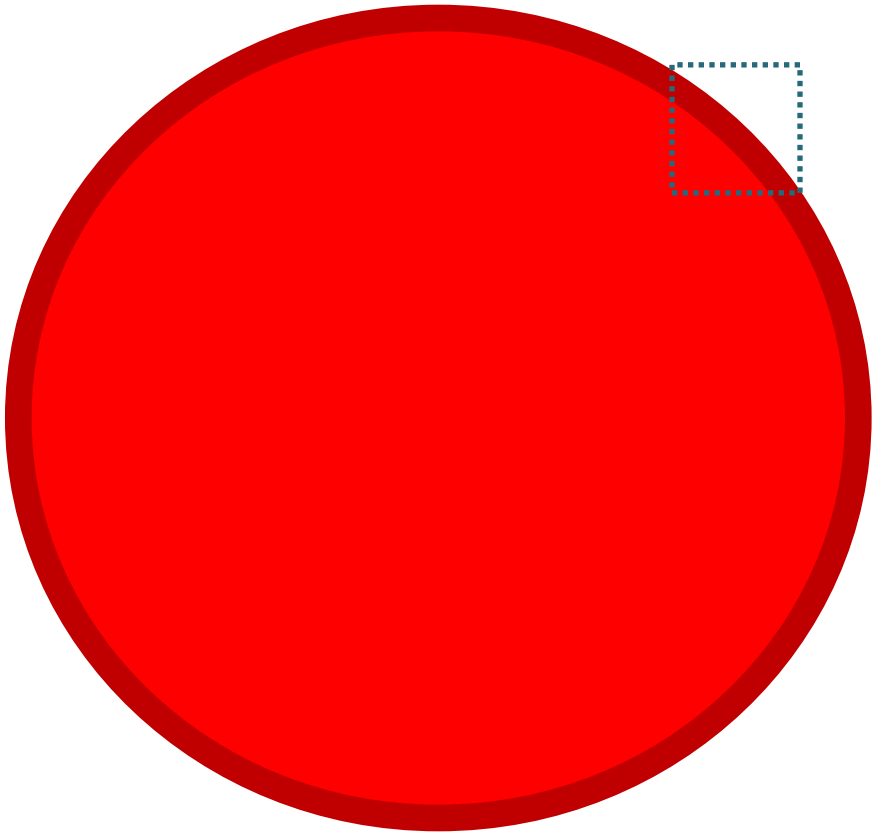
Yael Amsterdamer, Susan B. Davidson, Daniel Deutch
Tova Milo, Julia Stoyanovich, Val Tannen

Presented by Guozhang Wang

DB Lunch, Apr. 23rd, 2012

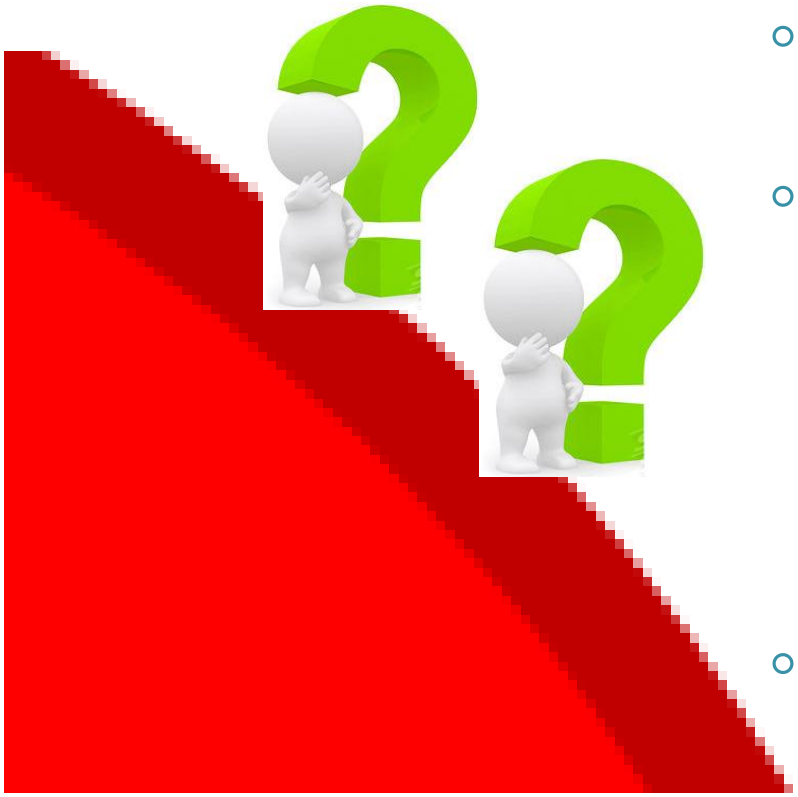
A Story of "How Research Ideas Get Motivated"

- A short time ago, somewhere in the Globe of CS Research …



Workflow Provenance

- Motivated by Scientific Workflows



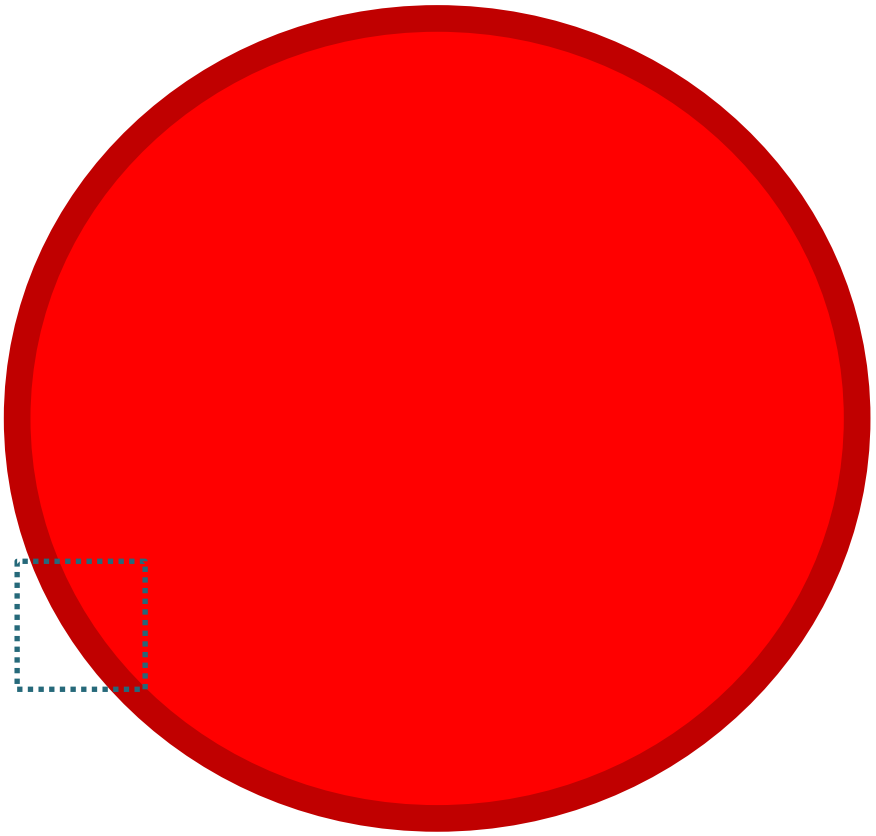
  - Community : IPAW
  - Interests: process documentation, data derivation and annotation, etc
  - Model : OPM

OPM Model

- Annotated directed acyclic graph
  - Artifact: immutable piece of state
  - Process: actions performed on artifacts, result in new artifacts
  - Agents: execute and control processes
- Aims to capture causal dependencies between agents/processes
- Each process is treated as a “black-box”

Meanwhile

- On the other side of the Globe …



Data Provenance (for Relational DB and XML)

- Motivated by Prob. DB, data warehousing ..

- Community: SIGMOD/PODS
- Interests: data auditing, data sharing, etc
- Model: Semiring (etc)

Semiring

- K-relations
  - Each tuple is uniquely labeled with a provenance "token"
- Operations:
  - • : join
  - + : projection
  - 0 and 1: selection predicates

A Datalog Example of Semiring

q(x,z) :- R(x, _,z), R(_, _,z)
q(x,z) :- R(x,y, _), R(_ ,y,z)

R

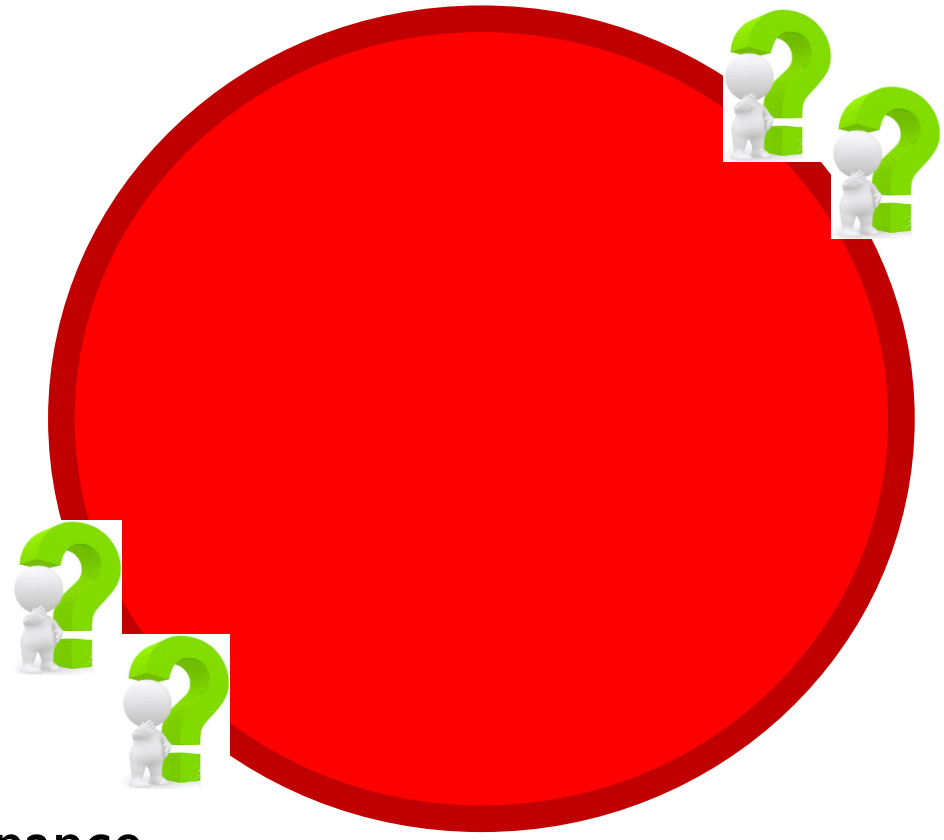
a	b	c	p
d	b	e	r
f	g	e	s

q(R)

a	c	$2p^2$
a	e	pr
d	c	pr
d	e	$2r^2 + rs$
f	e	$2s^2 + rs$

Slide borrowed from Green et al.

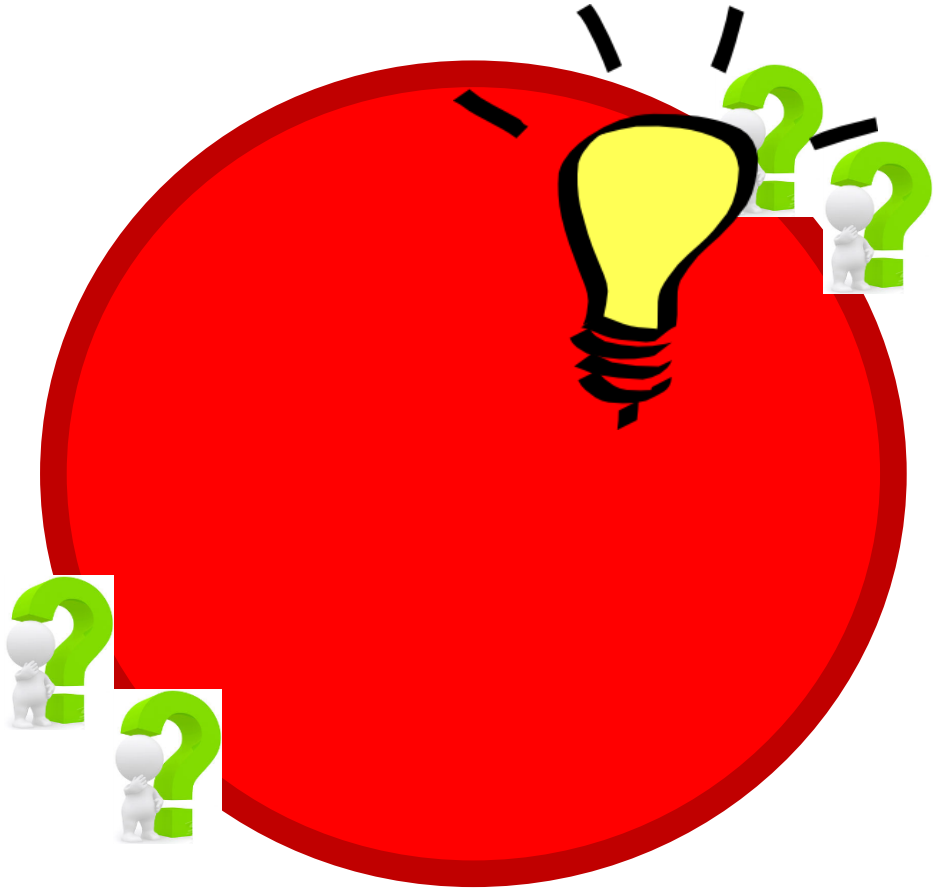
They Live Happily and Semi-Separately, Until …



Workflow Provenance Researchers

Data Provenance Researchers

Semiring Comes to Meet OPM



OPM's Drawbacks in Semiring People's Eyes

- The black-box assumption: each output of the module depends ***solely*** on ***all*** its inputs
  - Cannot leverage the common fact that some output only depends on small subset of inputs
  - Does not capture internal state of a module
- So: replace it with Semirings!

The Idea

- General workflow modules is complicated, and thus hard to capture its internal logic by annotations
- However, modules written in Pig Latin is very similar to Nested Relational Calculus (NRC), thus are much more feasible
- Let us write a paper, woho!

End-of-Story Disclaimer

This story is purely imaginative.

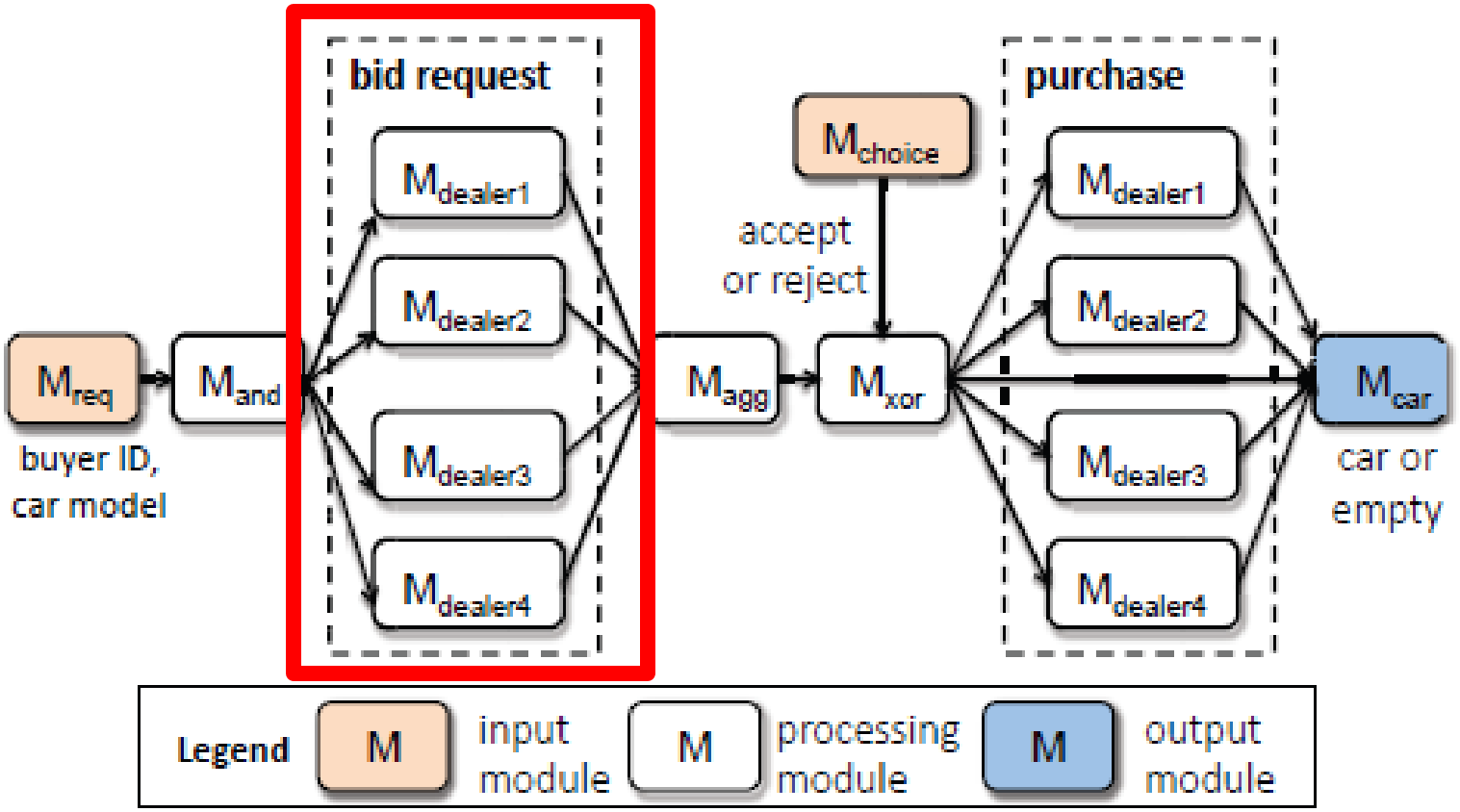
It is to be coincidental if there are similarities between the story and the real world.

Pig Latin

- Data: unordered (nested) bag of tuples

- Operators:
  - FOREACH t GENERATE f1, f2, … OP(f0)
  - FILTER BY condition
  - GROUP/COGROUP
  - UNION, JOIN, FLATTEN, DISTINCT …

Example: Car Dealership



Bid Request Handling in Pig Latin

Requests

UserId	BidId	Model

Cars

CarId	Model

SoldCars

CarId	BidId

InventoryBids

BidId	UserId	Model	Amount

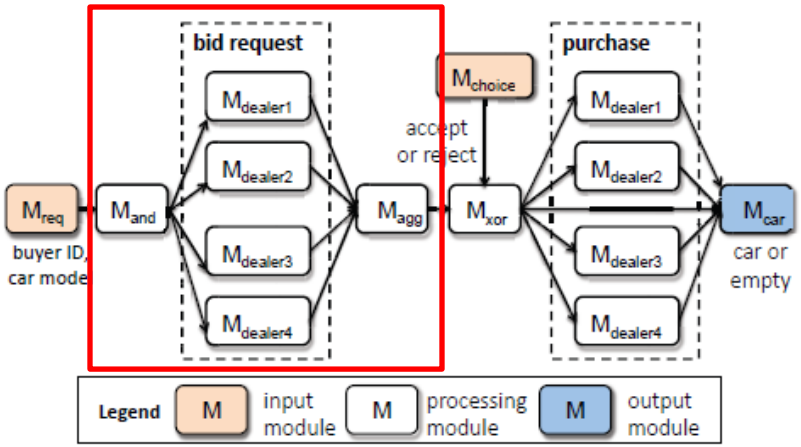
Bids

Model	Price

[illegible] NumA, NumS }

```
ReqModel = FOREACH Requests GENERATE Model;
Inventory = JOIN Cars BY Model, ReqModel BY Model;
SoldInventory = JOIN Inventory BY CarId,
                     SoldCars BY CarId;
CarsByModel = GROUP Inventory BY Model;
SoldByModel = GROUP SoldInventory BY Model;
NumCarsByModel = FOREACH CarsByModel GENERATE
      group as Model, COUNT(Inventory) as NumAvail;
NumSoldByModel = FOREACH SoldByModel GENERATE
      group as Model, COUNT(SoldInventory) as NumSold;
AllInfoByModel = COGROUP Requests BY Model,
                    NumCarsByModel BY Model,
                    NumSoldByModel BY Model;
InventoryBids = FOREACH AllInfoByModel GENERATE
FLATTEN(CalcBid(Requests,NumCarsByModel,NumSoldByModel));
```

Provenance Annotation

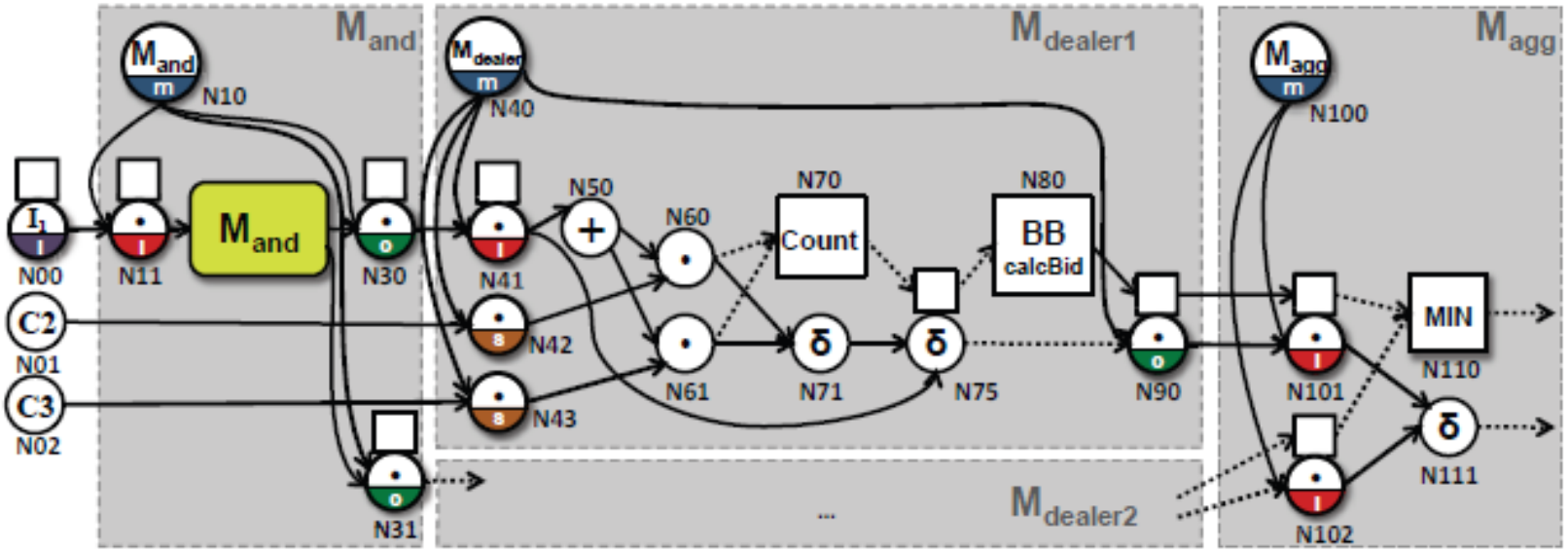


Cars

CarId	Model
C_1	*Accord*
C_2	*Civic*
C_3	*Civic*

Requests

UserId	BidId	Model
P_1	B_1	*Civic*

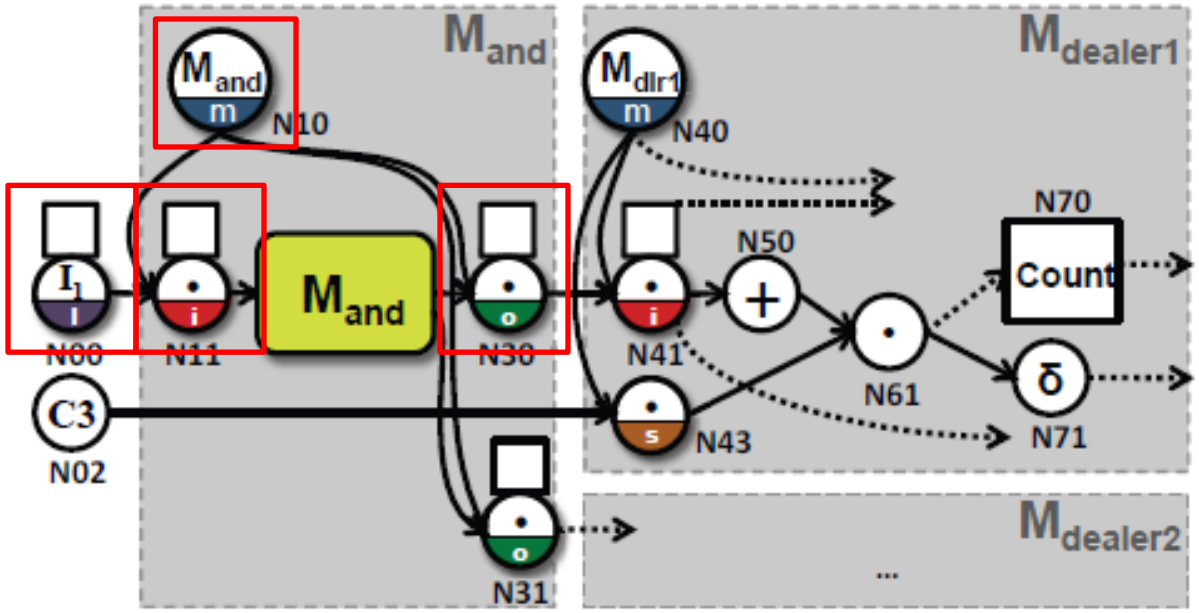


Provenance Annotation 1.1

- Provenance node and value nodes
  - Workflow input nodes
  - Module invocation nodes
  - Module input/output nodes

Requests

UserId	BidId	Model
P_1	B_1	*Civic*

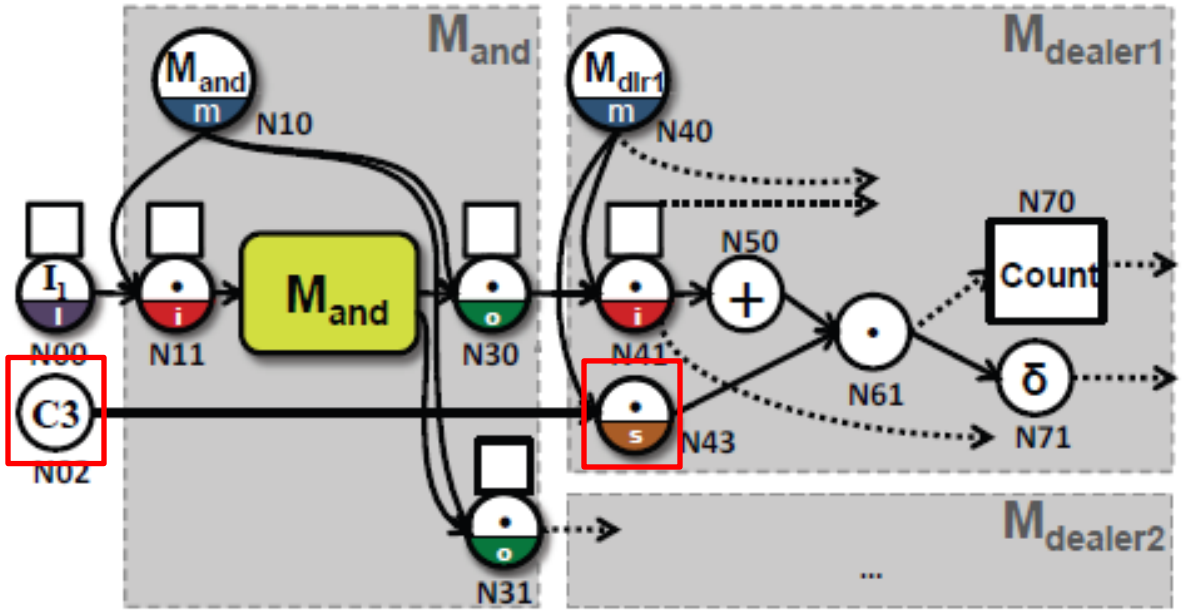


Provenance Annotation I.2

- State nodes
  - P-node for the tuple
  - P-node for the state

Cars

CarId	Model
C_1	*Accord*
C_2	*Civic*
C_3	*Civic*

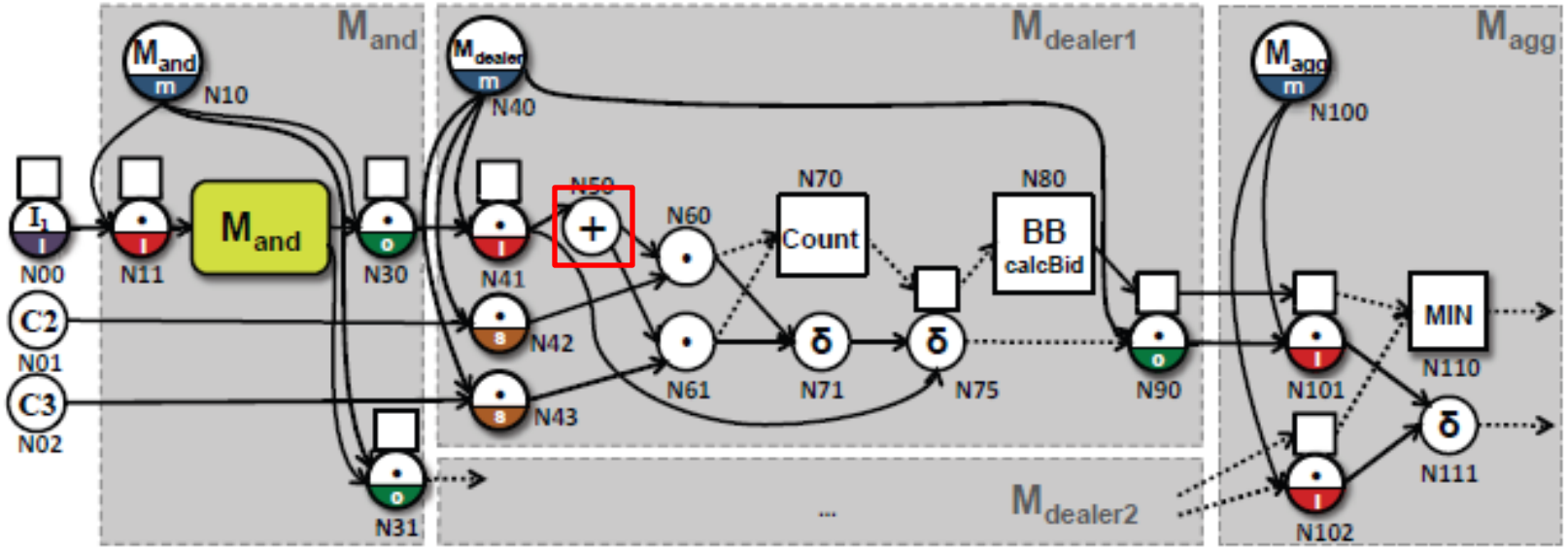


Provenance Annotation 2.1

- FOREACH (projection, no OP)
  - P-node with "+"

Requests

UserId	BidId	Model
P_1	B_1	*Civic*



Provenance Annotation 2.2

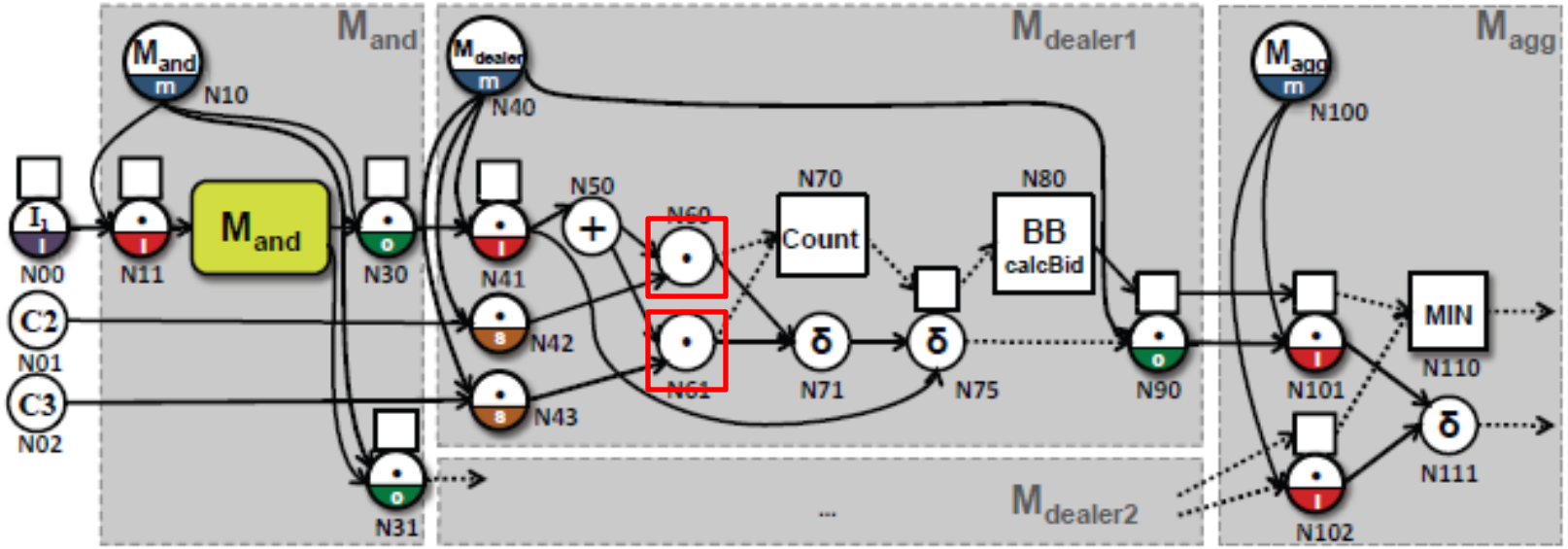
- JOIN
  - P-node with "*"

Requests

UserId	BidId	Model
P_1	B_1	*Civic*

Cars

CarId	Model
C_1	*Accord*
C_2	*Civic*
C_3	*Civic*



Provenance Annotation 2.3

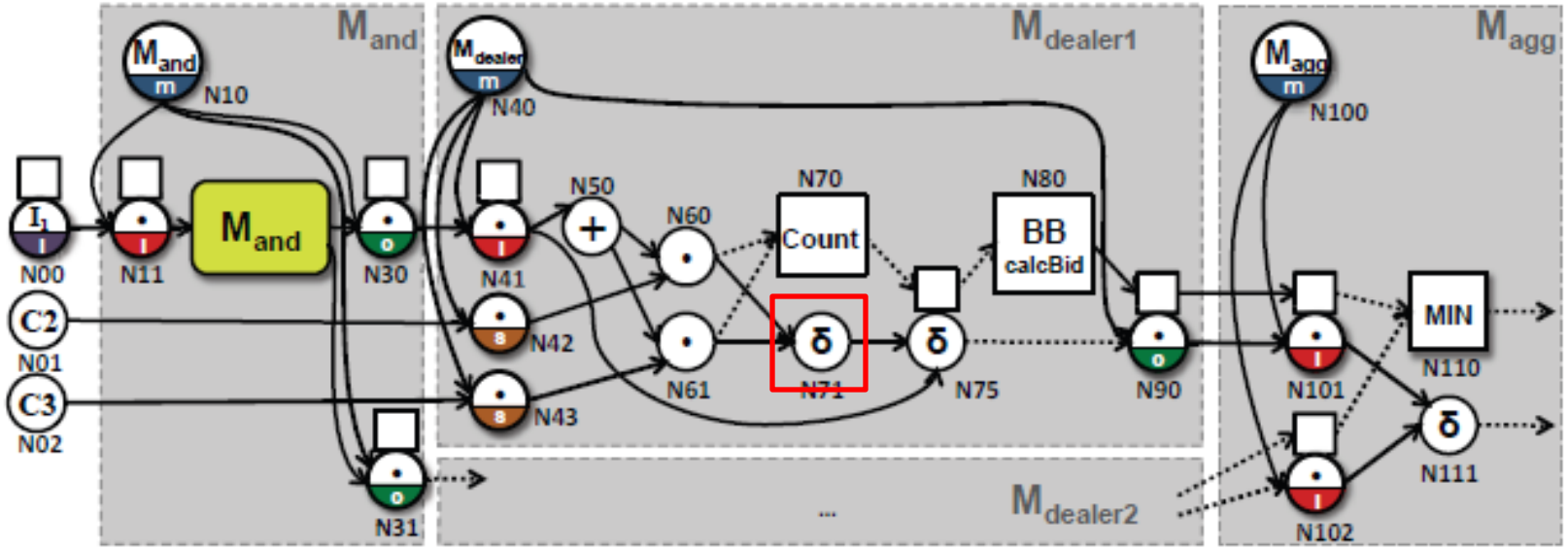
- GROUP
  - P-node with "∂"

Requests

UserId	BidId	Model
P_1	B_1	*Civic*

Cars

CarId	Model
C_1	*Accord*
C_2	*Civic*
C_3	*Civic*



Provenance Annotation 2.4

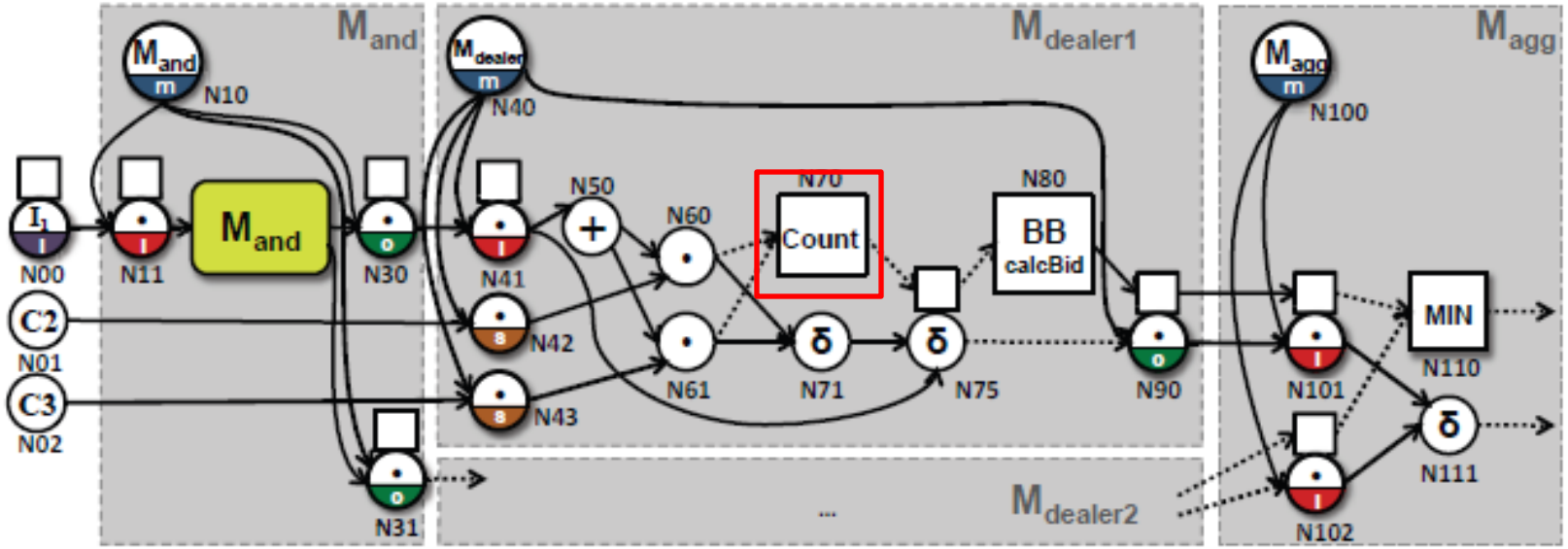
- FOREACH (aggregation, OP)
  - V-node with the OP name

Requests

UserId	BidId	Model
P_1	B_1	*Civic*

Cars

CarId	Model
C_1	*Accord*
C_2	*Civic*
C_3	*Civic*



Provenance Annotation 2.5

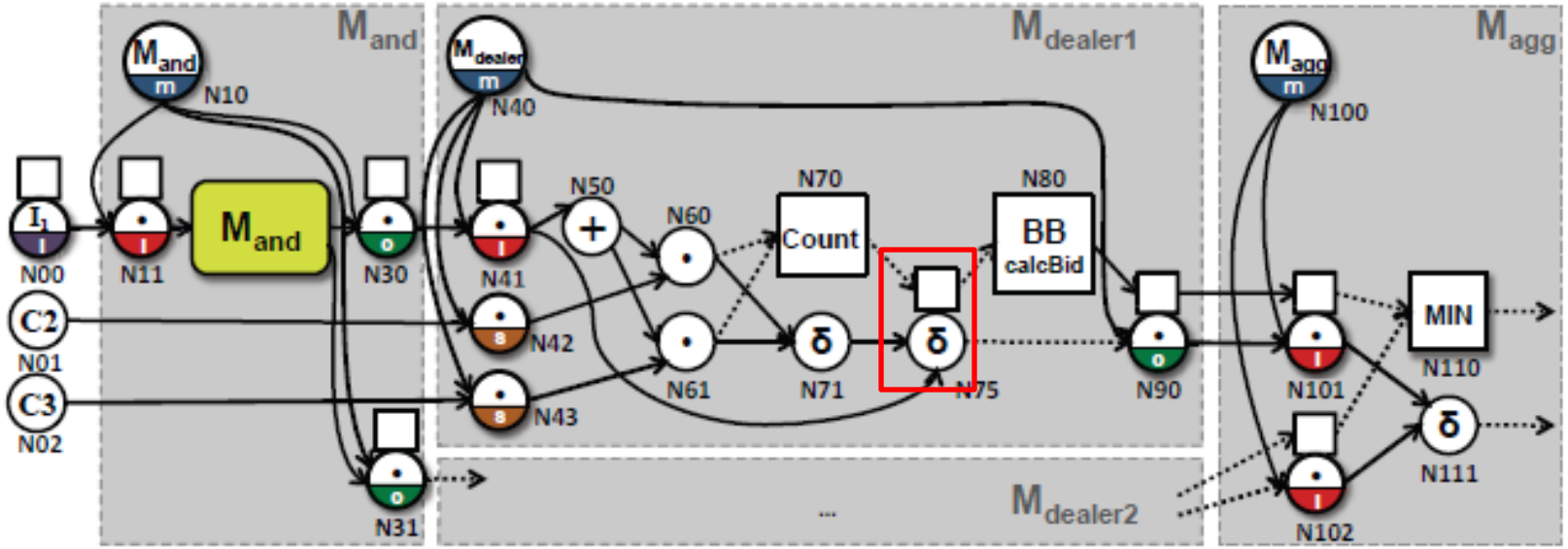
- COGROUP
  - P-node with "∂"

Requests

UserId	BidId	Model
P_1	B_1	*Civic*

Cars

CarId	Model
C_1	*Accord*
C_2	*Civic*
C_3	*Civic*



Provenance Annotation 2.6

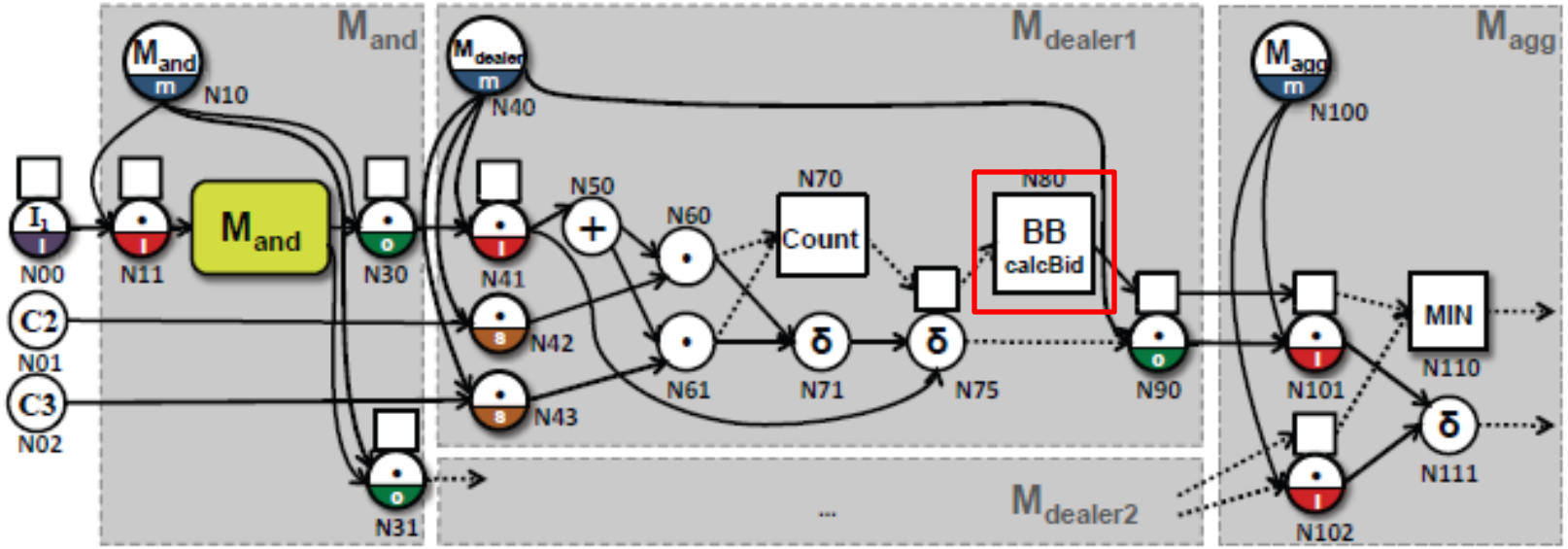
- FOREACH (UDF Black Box)
  - P-node/V-node with the UDF name

Requests

UserId	BidId	Model
P_1	B_1	*Civic*

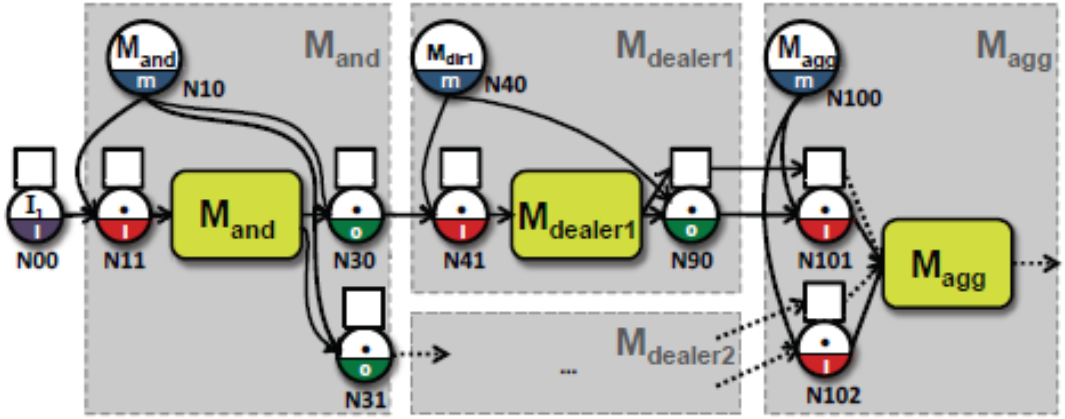
Cars

CarId	Model
C_1	*Accord*
C_2	*Civic*
C_3	*Civic*



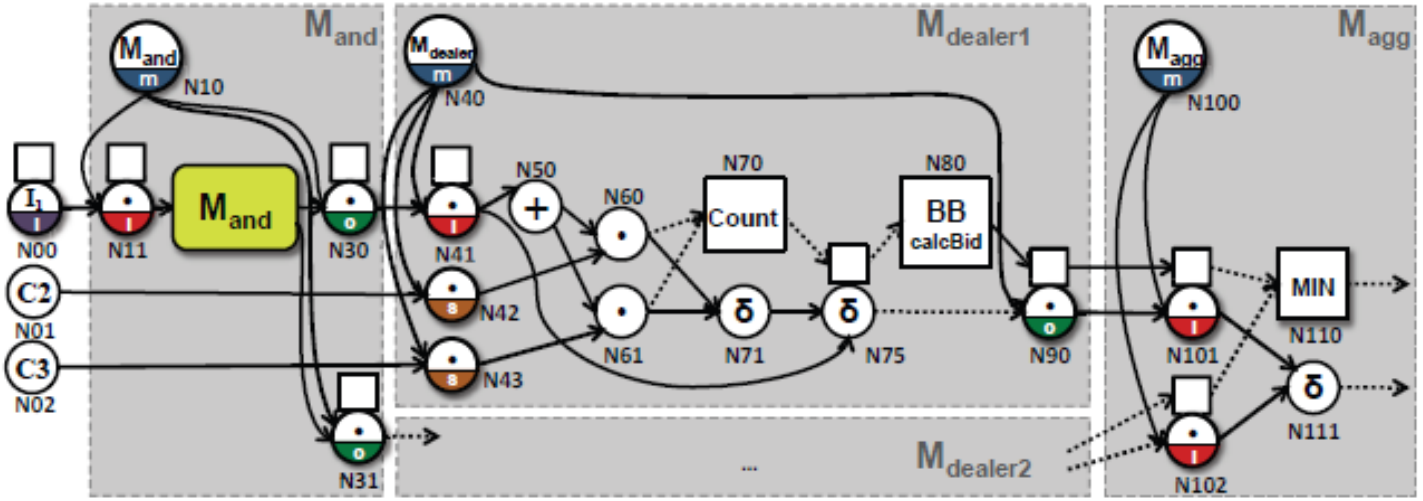
Query Provenance Graph

- Zoom-In v.s. Zoom-Out



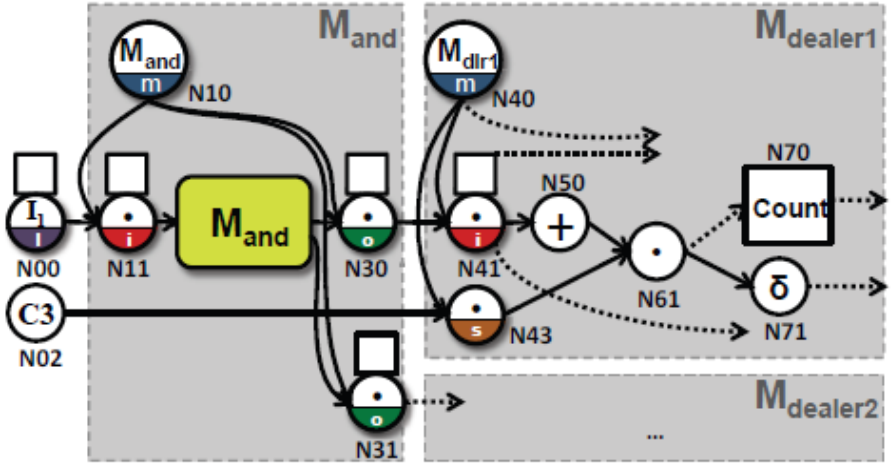
Coarse-grained

Fine-grained



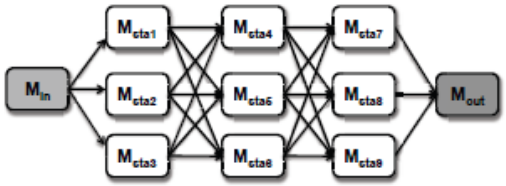
Query Provenance Graph

- Deletion Propagation
  - Delete the tuple P-node and its out-edges
  - Repeated delete P-nodes if
    - All its in-edges are deleted
    - It has label • and one of its in-edges is deleted

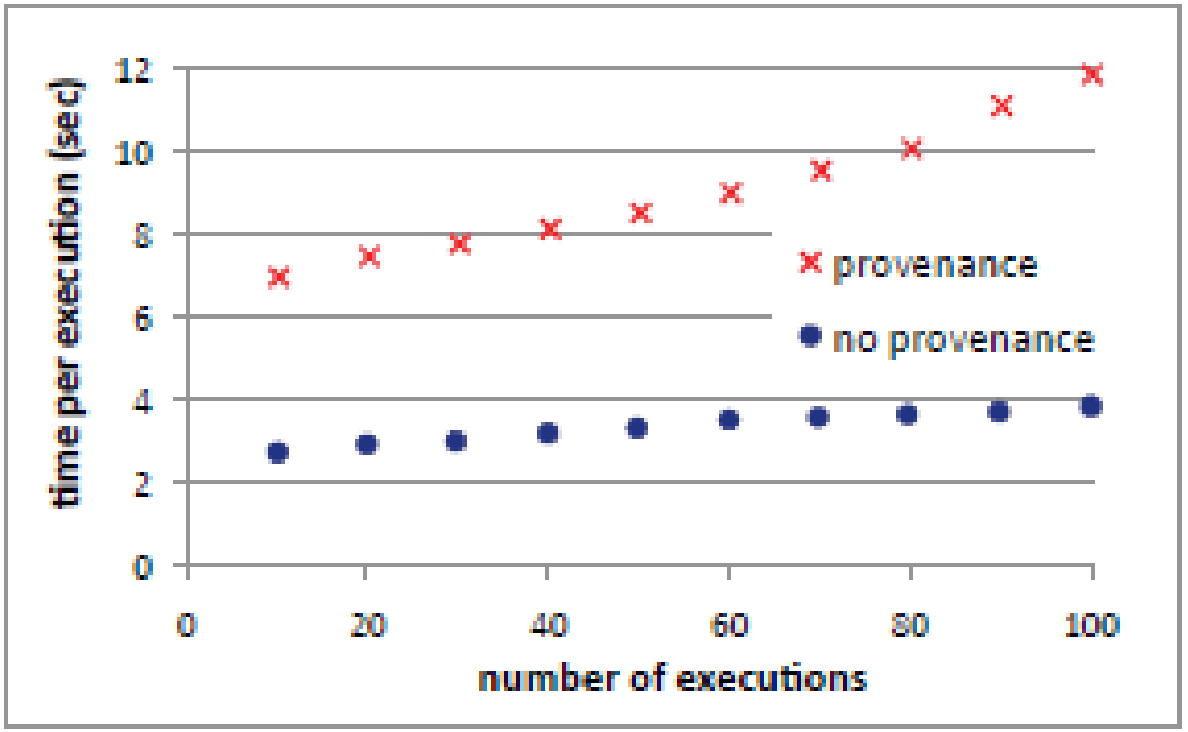


Implementation and Experiments

- Lipstick prototype
  - Provenance annotation coded in Pig Latin, with the graph written to files
  - Query processing coded in Java and runs in memory.
- Benchmark data
  - Car dealership: fixed workflow and # dealers
  - Arctic Station: Varied workflow structure and size



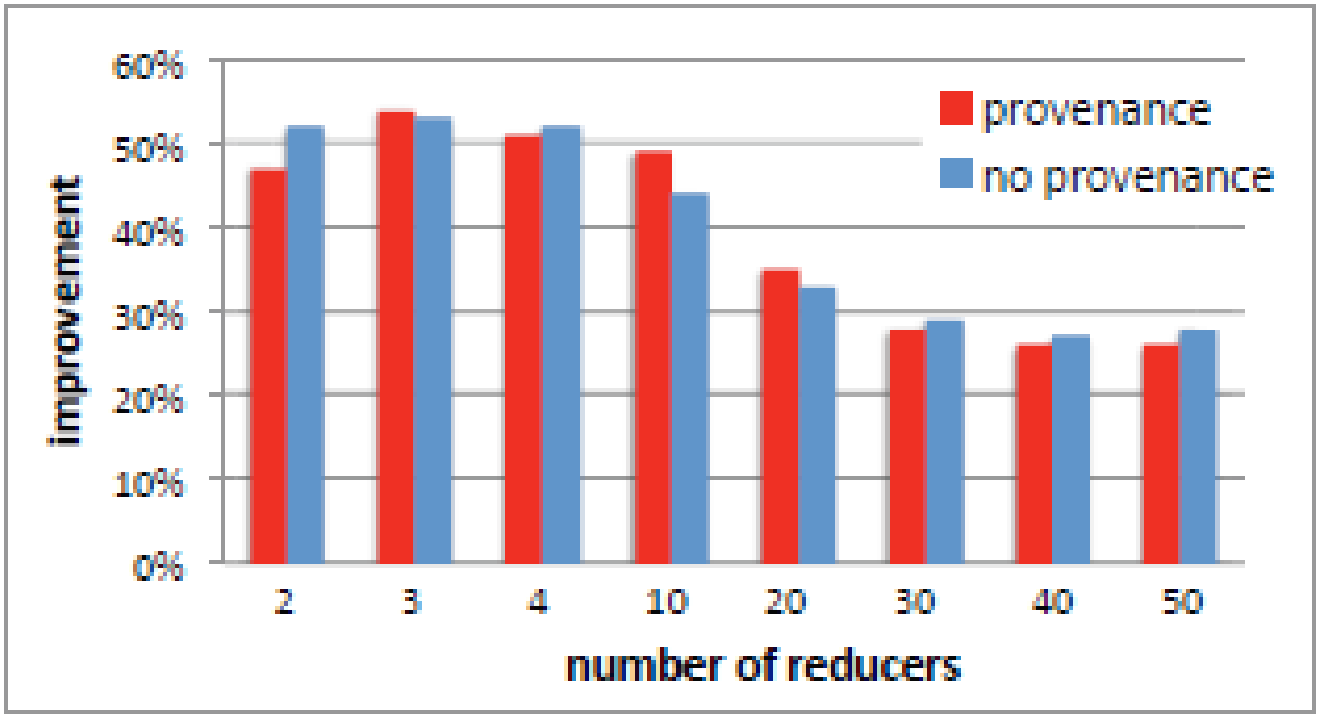
Annotation Overhead



(a) *Car dealerships*, local mode

- Overhead increases with execution time

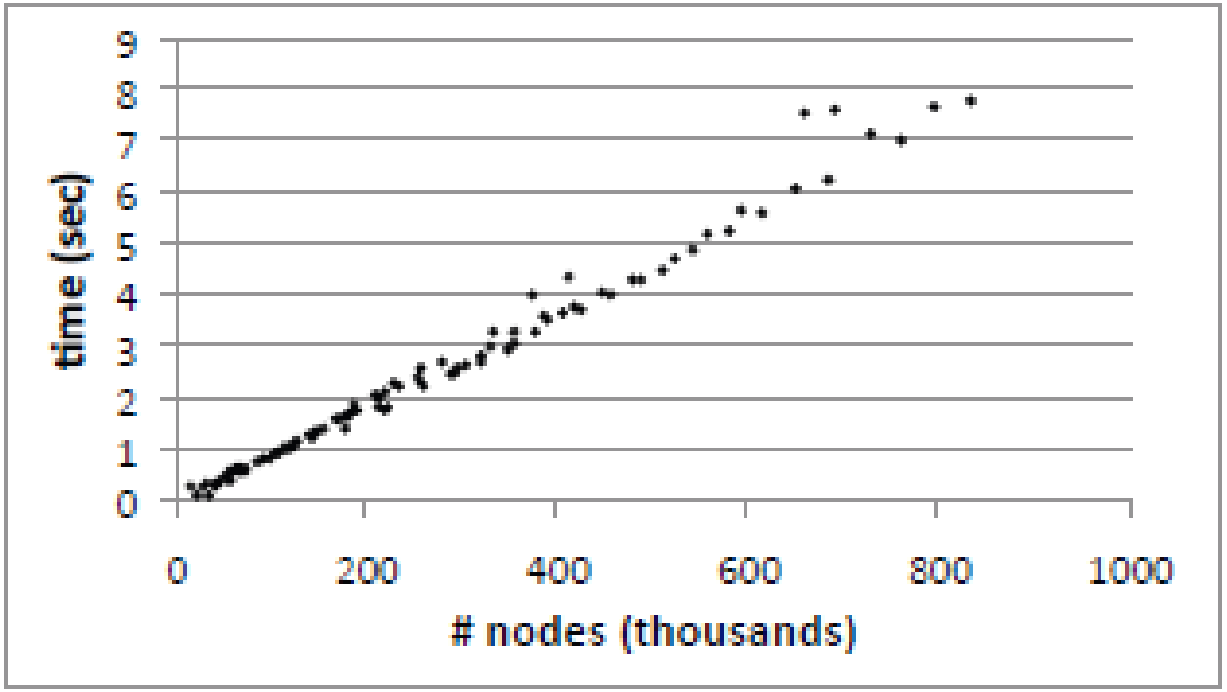
Annotation Overhead



(c) *Car dealerships*, impact of parallelism

- Parallelism helps with up to # modules

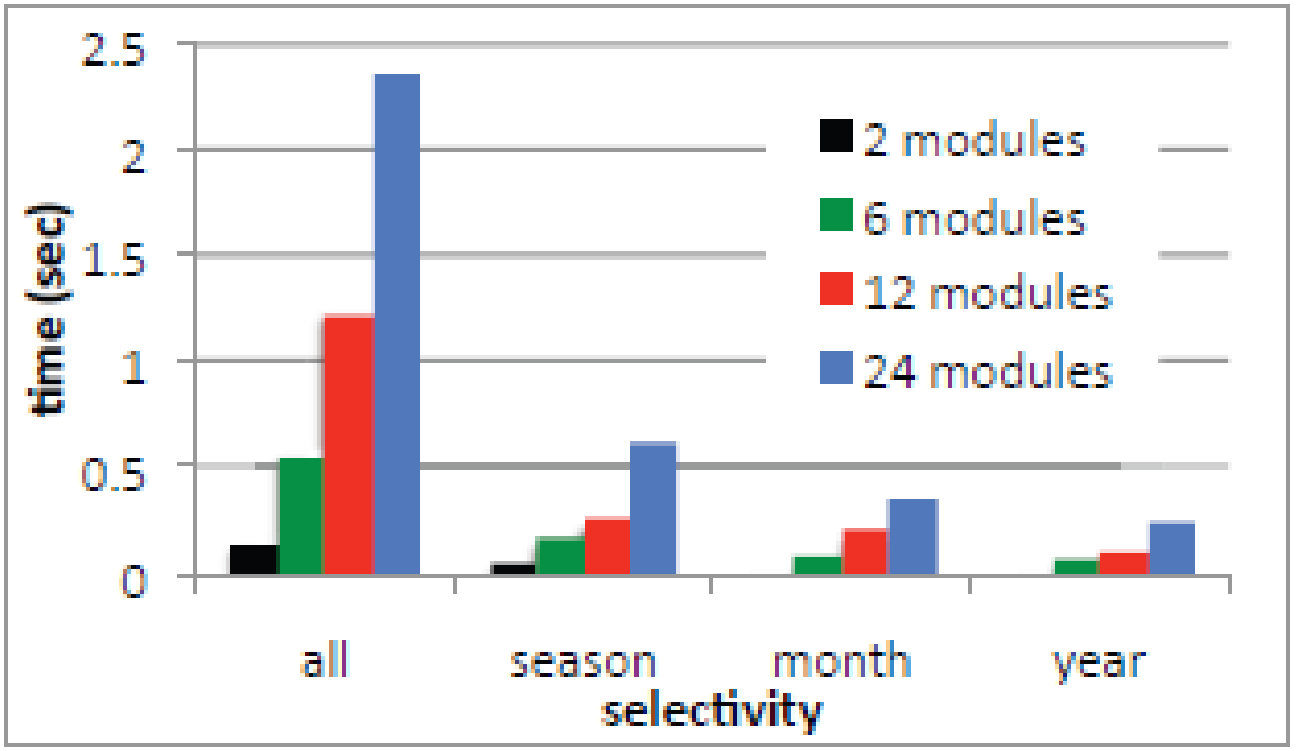
Loading Graph Overhead



(a) *Car dealerships*

- Increase with graph size (comp. time < 4 sec)

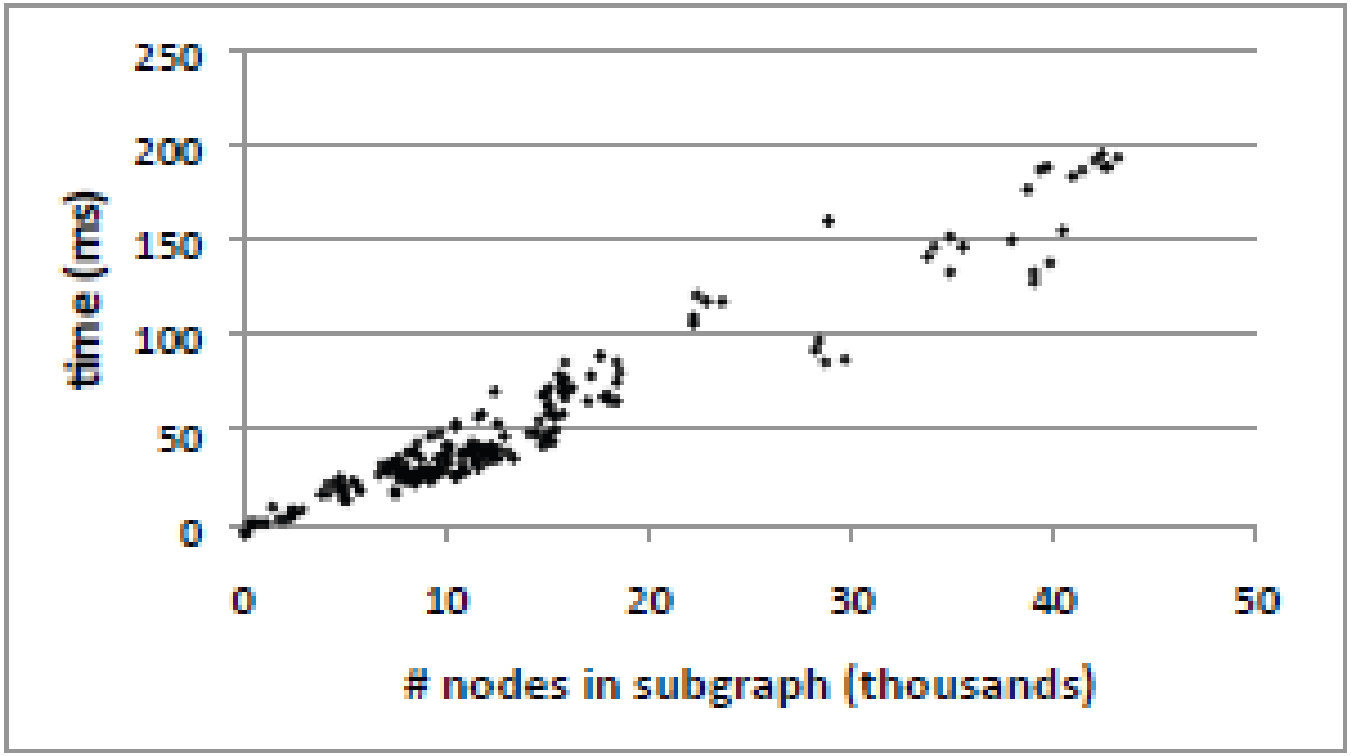
Loading Graph Overhead



(b) *Arctic stations* dense, fan-out 2

- Feasible with various sizes (comp. time ~ 8 sec)

Subgraph Query Time



(b) Subgraph, *Car dealerships*

- Query efficiently with sub-second time

Conclusions

Thank You!

- Data provenance ideas such as Semirings can be brought to workflow provenance for those "relational" programs

- No second conclusion, sorry ..

Backup Slides

- The introduction of MapReduce/Dryad/Hadoop …
  - Originally designed for data-driven web applications
  - Helped gaining DB researchers attentions back to workflow apps