Provenance in the Wild

TaPP '11

3rd USENIX Workshop on the Theory and Practice of Provenance

June 20–21, 2011 | Heraklion, Crete, Greece

usenix

Sponsored by USENIX in cooperation with ACM SIGMOD, ACM SIGPLAN, and the W3C Greece Office

Elaine Angelino, Marc Chiarini, John Lyle,
Margo Seltzer, Christina Strong

June 20, 2011

What’s the Problem?

- What does it mean to collect provenance when you don’t control:
  - The data (types, format, organization, structure)
  - The operators
  - The environment in which its processed
- Can you impose/ extract any semantic meaning to provenance when it’s collected by a herd of cats?

http://www.newsrealblog.com/wp-content/uploads/2011/04/Herding-Cats.jpg

What do the Cats do?

- They use data in arbitrary formats
  - Flat files
  - Unstructured, semi-structured, badly-structured
  - Proprietary formats
  - The cram twelve different kinds of data into a single container.
- Transformations are arbitrary code
  - Pick your favorite turing-complete language.
  - Apply said language to data.
  - Transformations can depend on the environment.
  - Repeat
- They move data around
  - Download objects from the web
  - Copy, rename objects
  - Replace objects
- They install new software
  - New programs
  - New libraries
  - New compilers

A Simple Example

- What is the provenance of LS1.OUT?

```
% cd ~margo/talks/tapp-dir
% ls –l > ~margo/LS1.OUT
```

- Audience Participation
- Given the following:

```
% cd ~margo/talks/tapp-dir
% ls –l > ~margo/LS2.OUT
```

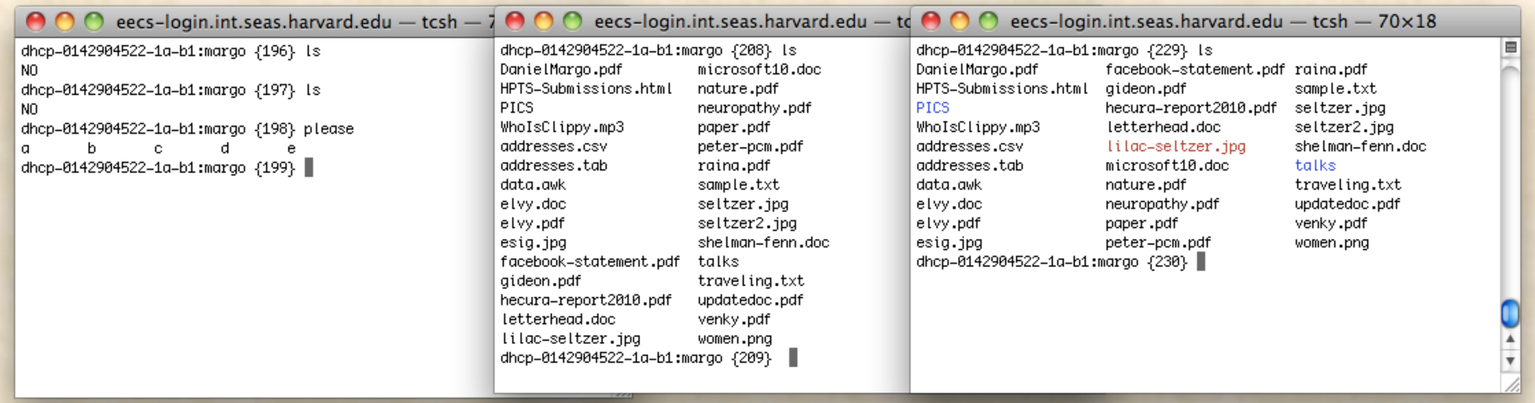
- Is the provenance of LS1.OUT the same as that of LS2.OUT?

Not so simple?

- What happened?
  - The behavior of `ls` depends on the environment.
- Who knew?
  - `ls` knew
  - The shell knew what its environment was
    - BUT – did not necessarily know that `ls` depended on it
  - The operating system knew
    - BUT – like `ls`, did not know that `ls` depended on it

Many other problems

- This is just one example of what can go wrong.
- Many others exist:
  - What program runs when a user types `ls`?
  - In what directory was `ls` run from?
  - What environment variables are set?


eecs-login.int.seas.harvard.edu — tcsh — 7

```
dhcp-0142904522-1a-b1:margo {196} ls
NO
dhcp-0142904522-1a-b1:margo {197} ls
NO
dhcp-0142904522-1a-b1:margo {198} please
a       b       c       d       e
dhcp-0142904522-1a-b1:margo {199}
```

eecs-login.int.seas.harvard.edu — tc

```
dhcp-0142904522-1a-b1:margo {208} ls
DanielMargo.pdf          microsoft10.doc
HPTS-Submissions.html    nature.pdf
PICS                     neuropathy.pdf
WhoIsClippy.mp3          paper.pdf
addresses.csv            peter-pcm.pdf
addresses.tab            raina.pdf
data.awk                 sample.txt
elvy.doc                 seltzer.jpg
elvy.pdf                 seltzer2.jpg
esig.jpg                 shelman-fenn.doc
facebook-statement.pdf   talks
gideon.pdf               traveling.txt
hecura-report2010.pdf    updatedoc.pdf
letterhead.doc           venky.pdf
lilac-seltzer.jpg        women.png
dhcp-0142904522-1a-b1:margo {209}
```

eecs-login.int.seas.harvard.edu — tcsh — 70×18

```
dhcp-0142904522-1a-b1:margo {229} ls
DanielMargo.pdf         facebook-statement.pdf raina.pdf
HPTS-Submissions.html   gideon.pdf             sample.txt
PICS                    hecura-report2010.pdf  seltzer.jpg
WhoIsClippy.mp3         letterhead.doc         seltzer2.jpg
addresses.csv           lilac-seltzer.jpg      shelman-fenn.doc
addresses.tab           microsoft10.doc        talks
data.awk                nature.pdf             traveling.txt
elvy.doc                neuropathy.pdf         updatedoc.pdf
elvy.pdf                paper.pdf              venky.pdf
esig.jpg                peter-pcm.pdf          women.png
dhcp-0142904522-1a-b1:margo {230}
```

What is the Fundamental Problem?

- Knowledge about what a program is doing is distributed among multiple entities:
  - The program itself
  - The environment
  - The operating system
  - What OS modules are located
  - The system libraries
  - The hardware
  - The data
  - ….

Provenance in a Multi-agent World

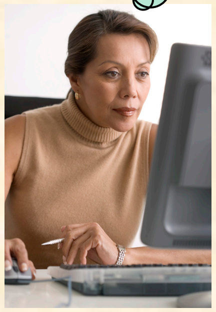
- Get over it: Accept the fact that multiple agents will have something to say about provenance.
- OK, but agents are cats! They:
  - Have different names for things.
  - Are interested in different kinds of objects (e.g., tuples versus files).
  - Have different types of transformations.
- Simply using a standard representation for multiple accounts doesn’t solve the problem.

Reconciling Accounts

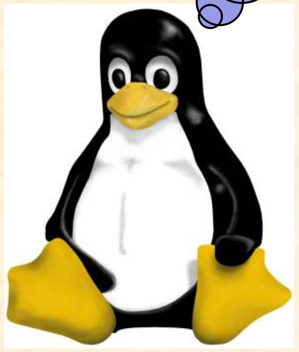
- Need to express a rich variety of relationships:
  - Identity
  - Containment
  - Instantiation
  - Composition
  - Versioning
- Each of these has a real, semantic meaning that queries need to exploit.

Reconciliation Example

I think I'll change all instances of filesystem to file system in orig.txt.



I see, you created a new file /tmp/xxx, then you renamed it, removing the file orig.txt



Let me put your changes in a new version of your file orig.txt

VI

http://www.aauwmi.org/state/SocialMedia/DiscussionForum/Computer_Woman.jpg

Implications

- Everyone needs to play in a provenance-aware world!
- Everyone needs to coordinate, but requiring that everyone use the same system is a losing proposition.
- Maintaining a provenance-aware commodity OS is a lot (a whole lot) of work!
- Provenance is grow-only; if everyone is collecting it, don’t we have a space problem?

Sustainability

- Maintaining our Linux-based provenance-aware kernel is not sustainable.
  - Linux kernel moves quickly; porting to new versions is hard, labor-intensive, and not research.
  - Staying on old versions makes the platform unattractive.
  - Solution: Can we develop an easier-to-sustain and more broadly accessible platform?
- Alternative:
  - Can we encapsulate everything we've learned in user-level libraries that applications, workflow engines, languages, etc can use?

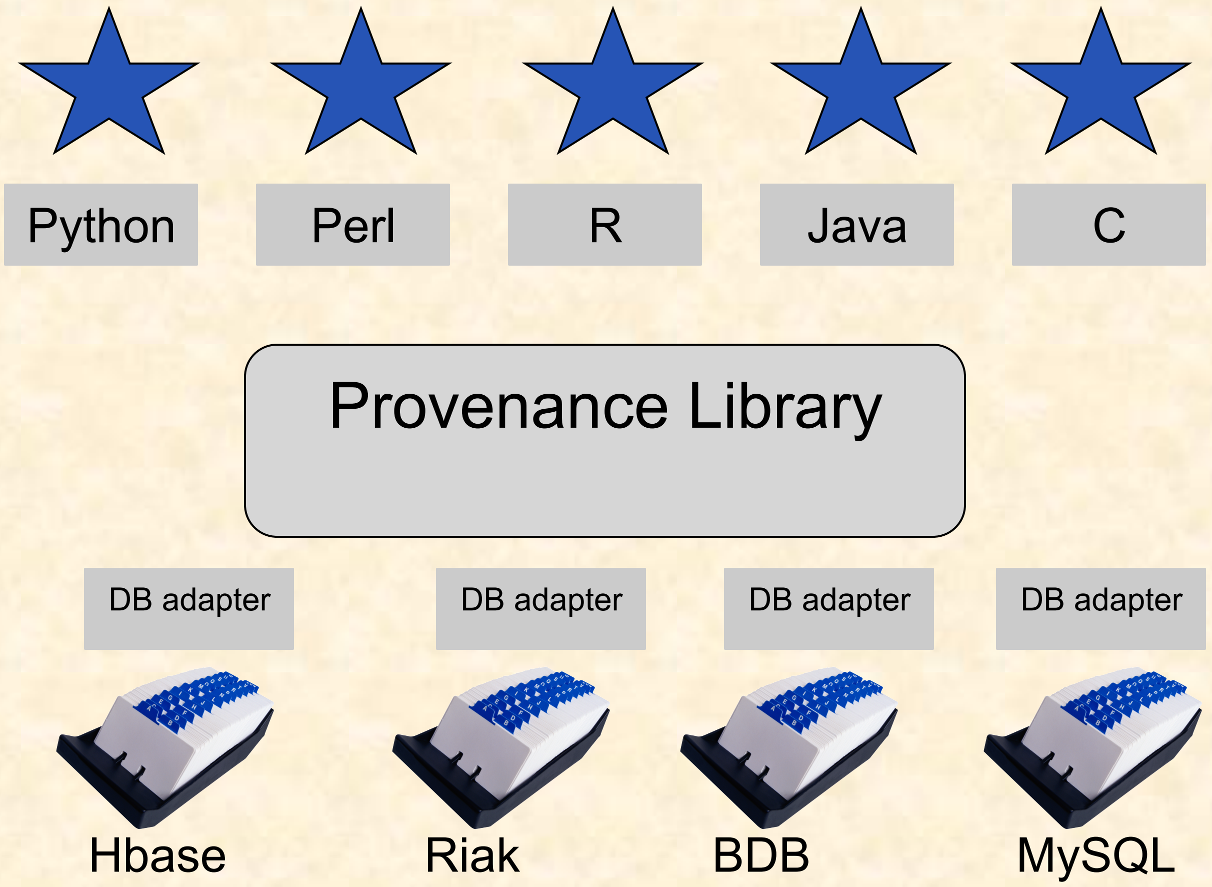
A Proposed Architecture

Applications
In multiple languages

Language adapters

Database adapters

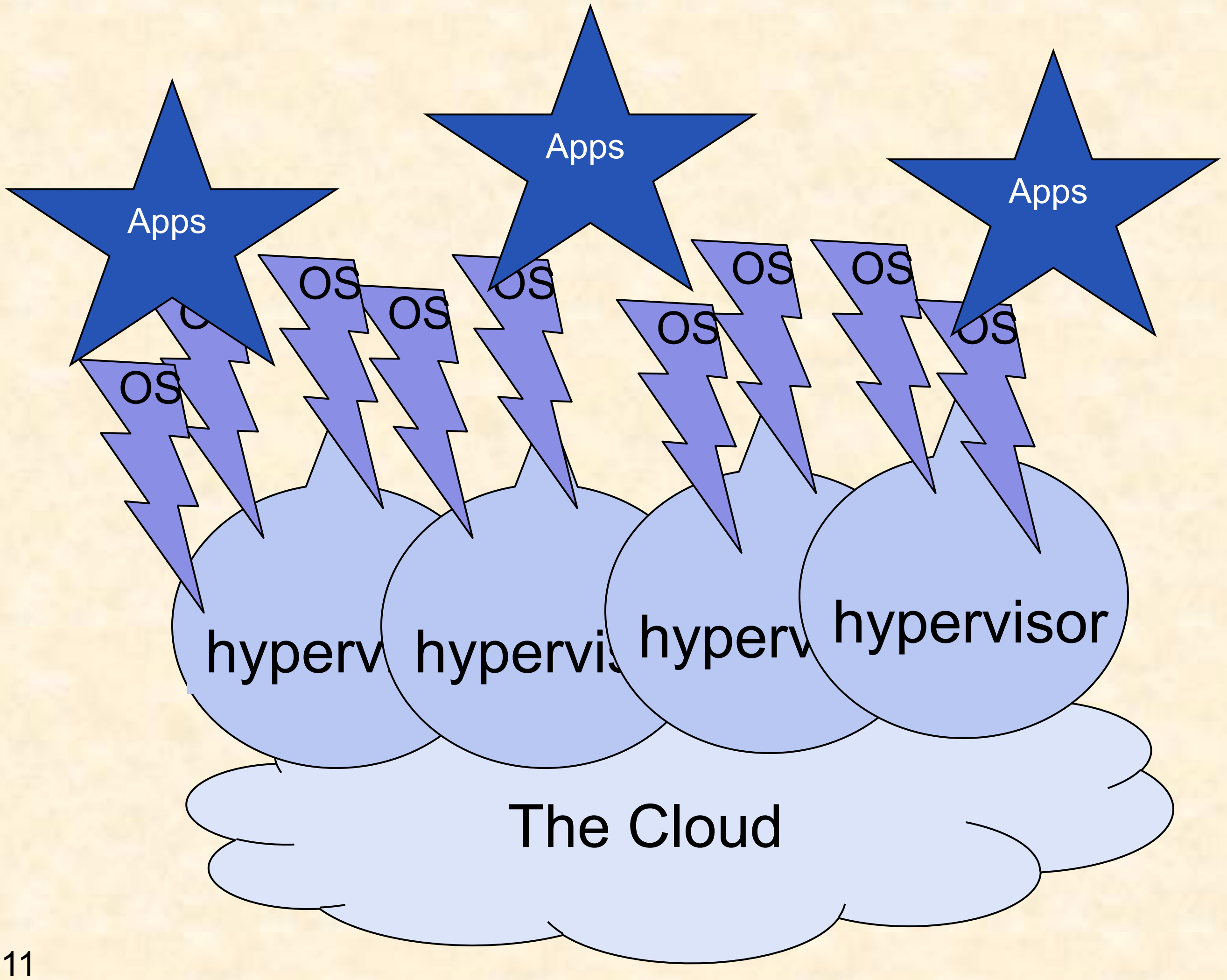
Provenance Store
With multiple implementations



About the Provenance Store

- Provenance is grow-only.
- There may exist some potential for pruning, but only for objects with no descendants.
- How do we manage provenance explosion?
- Compression!
    1. Apply web-graph compression techniques to provenance-graph compression.
    2. Look for common patterns, motifs, sub-graphs.
    3. Your good idea goes here.

Layers of Provenance



Specific Manifestations of these Problems

1. In integrating language and OS based systems: *Provenance Integration Requires Reconciliation*, **Elaine Angelino**
2. In collecting provenance in the hypervisor: *Collecting Provenance via the Xen Hypervisor*, **Marc Chiarini**
3. In the Cloud*: Challenges for Provenance in Cloud Computing*, **John Lyle**
4. In managing the scale of such data: *Compressing Provenance Graphs*, **Christina Strong**