Provenance Tipping Point

David Gammack
Marymount University
dgammack@marymount.edu

Adriane Chapman
The MITRE Corporation
achapman@mitre.org

Abstract

Capture is a known, difficult problem for provenance. Obtaining from the systems and programs exactly what happened has been a continuing struggle outside of database and workflow systems. The provenance research community has created libraries to log provenance, and has also embedded instances of capture agents within operating systems, specific programs, etc. However, it is impossible to know if we are inserting capture agents at both the optimal location and frequency in a given system for a high quality provenance graph. In this work, we develop an initial agent based model to simulate Activity and Entity interactions in a complex system of software. Using this model, we can attempt to define some generalized principles about type, frequency and distribution of provenance capture agents given a new system.

Categories and Subject Descriptors • Information systems ~Data provenance • Software engineering ~Entity relationship modeling

General Terms Management, Design.

Keywords *provenance, agent based models, simulations*

1. Introduction

The dream is for provenance to be a pervasive, always-on, everywhere information stream, in which data products can clearly be associated with their input and the processes that manipulated them. The capture of provenance information at every stage of data manipulation, from collection on ship through final publication, [4] achieves this dream in that it is a large system with a number of sub-systems and applications that are provenance-enabled. However, this is not the norm. Amongst many of the issues described in [5], one of the acknowledged pain points is “not enough provenance captured”. What level of involvement and provenance buy-in must we achieve before we tip from an “academic endeavour” to the vision of provenance everywhere?

When working with our government sponsors on their provenance needs, the initial set of questions are invariably:

- How many capture agents do I need to get good coverage of my system?
- Which systems should I create a capture agent for?
- What kind of capture agent should it be [manual, application, etc.]?

The answer to these questions is, of course, “it really depends on the set of applications within your system, the usage of the applications and how information usually flows through them,” in addition to the expected provenance uses. In order to truly answer this question, an analysis of the system’s operating environment and the intended usage for provenance is required. This work is intended to provide a partial answer: “in general, you need around X capture agents and should select the agents to optimize the following criteria.” Hopefully, it will assist system owners with a better “back of the envelope” calculation for expected number and type of capture agents needed to cover the system of systems (as a function of most capture for resource outlay).

In this work, we explore the “tipping point” phenomenon, i.e., the point where capturing provenance provides substantial utility for users. We do it by running simulations that estimate the provenance captured in a system of systems and varying numbers and types of capture agents to find the tipping point after which the provenance information is deemed “useful”. This initial work develops the model and provides a first set of results to prove the concept. The model can be later extended to provide a more in-depth analysis of the factors involved in the tipping point and refined to mimic a specific system of systems and possible capture agents.

Using NetLogo [19] we build an Agent Based Model (ABM) of provenance capture of various types, and different assumptions of likelihood of capture. We then exercise this model across thousands of simulations. We begin with four basic options for capture: manual, high-value coordination points, application-based, and do-nothing. It is cheaper to select manual and do-nothing as the initial approaches, but those techniques provide patchier provenance information. We simulate the provenance capture through a series of systems made up of several applications that pass data around. The model we develop in this work estimates the value of the provenance, and uses the cost of implementation as a guide to the number and types of capture agents to be deployed within a system of systems.

We begin by discussing related work in Section 2, in particular, the provenance capture agents that exist today and how they map to the types of capture agents in this work. Additionally, we provide background on the agent based modelling system that we will be using for our experimentation. In Section 3 we provide a brief motivation by looking at three different analytic cells who wished to use provenance, and the variation and conundrums inherent in this problem. In Section 4, we describe our model and present our implementation and initial results in Section 5. Because this is initial work, in Section 6 we describe extensions to the model and execution for later evaluation.

2. Related Work

2.1 Provenance

When provenance was limited to workflow systems or databases, provenance capture was a known and well contained problem. However, as we expanded outside of these systems to capture provenance in any system, the capture agent problem became serious. Creation of provenance capture libraries [8, 11] allows for the easy interface of a provenance capture agent with a provenance store, but the need to build and deploy a provenance capture agent still remains. We break down types of capture agents into: manual, application-based and coordination-point, as described in [6].

Manual: Many standards, such as [1], include provenance as components of the required metadata; in many instances, much of that information is populated by hand by a data curator. In general, we believe that the manual approach should be used sparingly, since it places a great burden on data users and cannot scale. Of particular interest are hybrid approaches in which the application itself is somewhat provenance-enabled, but the user makes the final decision as to what is important and needs to be stored, such as [10].

Application-based: In some cases, an application is so heavily used that it is beneficial to expend the resources to capture provenance information from just that application. Examples include [3, 7, 12].

Coordination-points: In many systems of systems, there are obvious "coordination points". These are resources that many users utilize, or that actively help order, transmit and manage data and jobs. Examples that have provenance capture agents today include MapReduce [15], UNIX kernel [13], GIT [14], ESB [2]. Provenance can be captured at these points by monitoring data flows.

For now, we will ignore provenance-introspection work, such as [18], in which capture agents are not needed and the provenance is inferred from the data itself. We believe that this work is essential for the completion of a provenance graph, but should be supplemental to capture agents until it fully matures.

The labour to provenance-enable is typically similar for an application or a coordination point. However, the outputs of each provenance capture agent are significantly different. Manual capture agents only produce provenance when the user is motivated; application based capture agents produce provenance for that specific application, while coordination points can "see" the provenance for many applications, jobs and data that pass through its borders.

2.2 Agent Based Models

Agent-Based Models (ABM) are computational descriptions of real world phenomena that we use to model and execute our simulations. The two main components to an ABM are the environment and its agents. The environment is the space in which phenomena occur, in our case an organization of interconnected process or systems. Agents, in this case Prov-DM "Activities" inhabit the environment. The designer of an ABM makes assumptions about the interactions between individual agents and between the agents and their environment. These interactions could be logical (if x then y), probabilistic (there is an x% chance of y) or function-based (y = f(x)). Once the interactions have been coded, simulations can be run and conclusions drawn. For a full introduction on ABM, see [9, 16]. NetLogo [19] is an agent-based modelling tool that allows the user to specify how agents (people/cells/objects) interact with each other and their environment and to then examine how the system evolves over time.

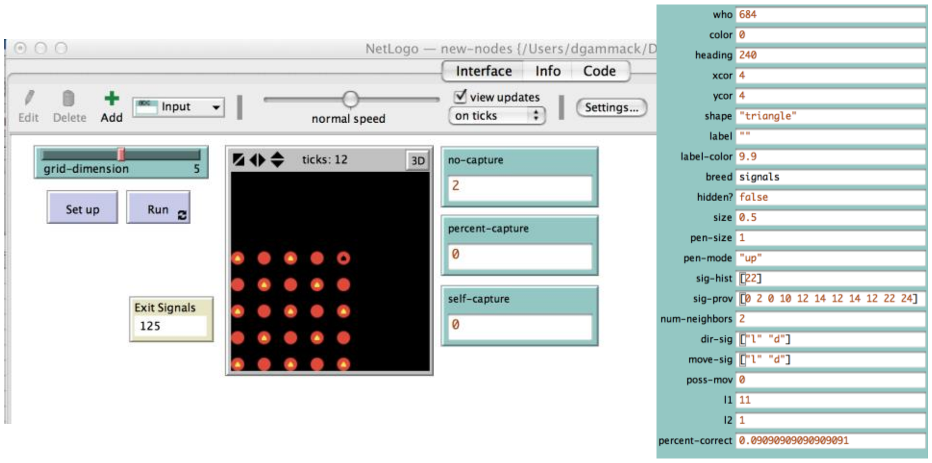
3. Project Characteristics

Table 1 highlights the real difficulty of convincing people to convert to an automatic provenance capture process. Each of the analytic groups is required to know what data they are using where, and approached the MITRE provenance team for insight into collecting provenance. For instance, A-Group[1], which deals with aviation analysis, is allowed to publish results that have been aggregated and anonymized sufficiently; thus the analyst must understand what data came from which system, and if enough sources were included to anonymize the information. N-Group is required to delete every instance of data, or any data derived from an external partner should that partner withdraw. Finally, the Office of Financial Research, within the US Department of the Treasury needs data and algorithm discoverability and classic scientific "means and methods" [17]. Each analytic group has a unique set of "favorite tools". None have a standard workflow, or required sequence of

Table 1: Characteristics of Analysis projects interested in tracking provenance

	# External Partners	**# Input Datasets**	**# Analysts**	**# Activities**	**# Intermediate Entities**	**Top identified tools used**	**Provenance Captured**
N-Group	5-10	50	10-12	>1000	Unknown	PostgreSQL, python	Manual – As New Files from external partners arrive Automatic – intermediate files get created on the shared drive Manual – relationships (what created the intermediate files)
A-Group	10	~400	~100	100-1000	Unknown	Oozie, MapReduce, HDFS	Files are manually named with an extension .x.y.z. X indicates a schema change, Y an algorithm change and Z upstream data change, so that output files can be associated with the right schema, algorithm and input data.
Office of Financial Research	100s	Unknown	100s	>1000	Unknown	R, Java, Matlab	Individual's notes

[1] MITRE policy requires us to anonymize the names of these projects.



view updates ✓
on ticks
grid-dimension 5
ticks: 12
no-capture 2
percent-capture 0
self-capture 0
Exit Signals 125

who	684
color	0
heading	240
xcor	4
ycor	4
shape	"triangle"
label	""
label-color	9.9
breed	signals
hidden?	false
size	0.5
pen-size	1
pen-mode	"up"
sig-hist	[22]
sig-prov	[0 2 0 10 12 14 12 14 12 22 24]
num-neighbors	2
dir-sig	["l" "d"]
move-sig	["l" "d"]
poss-mov	0
l1	11
l2	1
percent-correct	0.09090909090909091

a) b)

Figure 1: NetLogo session mid-simulation: a) basic setup of tool; b) inspection of "data agent" 684 and the provenance that was collected by the capture agents (sig-hist) vs what really happened to the data (sig-prov).

steps and tools; that is, if an individual analyst wishes to work in an entirely non-standard language or data store, they may. Input and intermediate data is freely shared and reused between analysts. In other words, each project environment is unique in their tools, data content, and information flow.

Interestingly, the N-group has made the most in-roads to provenance capture. Of all of the analytic projects described, this is the newest, and the individuals standing it up inquired about provenance from the get-go. While no funds were set aside to actually create provenance, they were willing to create a long-term strategy to begin incorporating provenance. This included: 1) determining the information required in the provenance for their needs; 2) forcing analysts to manually populate this information; 3) use the resulting "poor provenance store" and "save analyst time" argument to identify resources and requirements for greater automation. This "wait and see" attitude was partially due to funding and time constraints, but was also because at the initial onset, it was impossible (without seeing how the analysts were using which tool) to give a rough assessment of how many capture agents of which type were needed. Hence, we wish to find a way to determine a "rough estimate" of capture agents needed given the size of a particular system.

4. The Model

As this is the initial attempt at using a model to estimate provenance capture agent needs, we have begun as generically as possible. Because we ultimately want to determine whether the provenance is useful, we must first define "Good Provenance". There are several choices for this, depending on the actual usage of the provenance, including:

- 100% of all provenance is captured
- 80% of all possible provenance is captured
- At least 1 complete path between a source and sink exists

We have chosen to take the "80%" definition of "Good Provenance" for this work since we feel that 100% is completely unrealistic. However, the model is adaptable, and can be run with any of these options.

Additionally, the following components are also modelled: processes, data flow and capture agents. These components are modelled as follows:

- Processes accept data in, and they output data.
- Data is introduced at a "source" process, and is removed from the system when it reaches a "sink" process after it flows through the intermediate processes.
- Capture agents are of type: manual, application-based and coordination-points.

For this set of simulations, a process must output data to at least 1 other process that it is connected to (which one it goes to is probabilistic). After that, it is a degrading probabilistic function that the data is also output to any of the other connected processes. For example, a process could be: an ESB, a Hadoop system running MapReduce, a database system, etc. Data flows from process to process and can be split (a process takes in 1 piece of data and outputs to 2 different processes).The capture agents are randomly distributed over processes and have the following charateristics in this simulation:

- *Manual:* Because we are particularly sceptical of user's creation of provenance, we ran our simulations with the provenance being recorded 10% of the time.
- *Application-based* agents will always capture the provenance for the process they are attached to.
- *Coordination-based* agents will capture the provenance for the process they are attached to and all of the processes that are connected to them.

We have intentionally not defined our model to look like any pre-configured system of systems, nor have we assigned the provenance capture agents with any design as to what systems would "normally" reside at a particular point in a system of systems. Our experience, with just the three systems described above is that there is no norm.

5. Implementation and Evaluation

The NetLogo environment describes a system as a set discrete areas called patches. An agent in an ABM is an autonomous "individual" that reacts to its environment. Each agent in this implementation represents a process in our model. For simplicity sake in this model, we place an agent representing a process from our model on each patch in the environment. Additionally, an agent that interacts with other "process" agents represents the data in our model. For our initial simulation, a grid of agents, each agent is connected to the agents on each side. Figure 1, we show a run with 25 processes (red

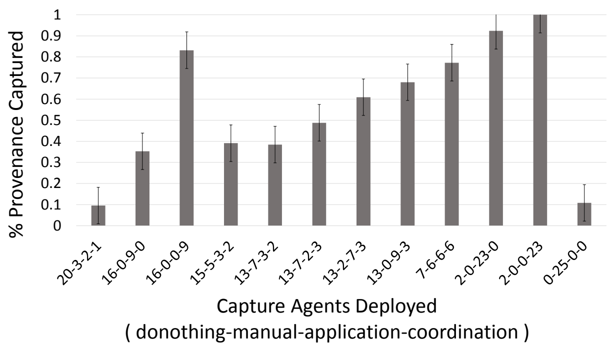


Figure 2: Simulation results altering the number and type of provenance capture agents shown with decreasing numbers of "do nothing" capture agents, and the result on % captured.

circles); triangles indicate a data flow is currently moving through that process.

Now that we have a model, and a NetLogo implementation of it, what does it tell us about the quality of the provenance, and how many capture agents are actually needed?

A back of the envelope calculation shows that in the 25 process model we run in the simulations, the minimum set of capture agents required for full coverage is: 6 coordination points and 2 application based. Figure 2 shows how the number and type of capture agents impacts the usefulness of the provenance captured. Each bar is labelled by the number of each type of capture agents spread throughout the system. For instance, 13-7-2-3 means there are 13 "do nothing" (no provenance captured), 7 manual, 2 application-based and 3 coordination-point-based. Of note, even though a total of 8 capture agents can theoretically be deployed to capture 100% of the provenance, because we cannot necessarily find or deploy the provenance capture agents in the "best" place, on average only 80% of the provenance is captured with 9 capture agents. Interestingly, although 16-0-0-9 has more "do nothing" nodes (16) than many of the other runs, it isn't until 7-6-6-6 (7 do nothing, 6 manual, 6 application based and 6 coordination-points) that the utility of the provenance is close. Thus, large increases in both manual and application-based capture are needed to compensate for minor decreases in coordination-based capture.

6. Future Work

In this work, we outline the initial problem statement and generate the first models for system interaction and provenance capture. One potentially difficult aspect of modelling is deciding what is crucial information and what can be neglected. In this initial work, we created a simplistic model that can answer basic questions about provenance capture agent numbers and types; we wish to expand the model to include a fuller evaluation of:

1. Cost of provenance capture agent to determine what sort of capture agents are produced (or not)
2. Determination of how signal propagates through system
3. System specification
   a. Specific connections of applications within the system
   b. Number of paths through the system
   c. Size
   d. Query only systems (do not pass signal through, but return answer to initiating system)
4. What is determined "complete" provenance.

Using these models, we hope to garner a better understanding of how many and what type of provenance agents are likely to be deployed in a system of a particular size. The models can be analysed via Latin Hypercube Sampling with Partial Rank correlations to determine the best fit of capture agents given a particular cost expenditure. We hope that this better understanding will assist in conversations around, "I want to provenance enable my system of systems, how painful is it likely to be?"

References

[1] "North American Profile of ISO19115:2003 - Geographic Information - Metadata." NAP Metadata Working Group 2005.

[2] M. D. Allen, A. Chapman, B. Blaustein, and L. Seligman, "Provenance Capture in the Wild," in *IPAW*, 2010.

[3] H. U. Asuncion, "Automated data provenance capture in spreadsheets, with case studies," *Future Generation Computer Systems*, vol. 29, pp. 2169-2181, 2013.

[4] P. C. Brauer, A. Czerniak, and W. Hasselbring, "Start Smart and Finish Wise: The Kiel Marine Science Provenance-Aware Data Management Approach," *Theory and Practice of Provenance*, 2014.

[5] J. Cheney, S. Chong, N. Foster, M. I. Seltzer, and S. Vansummeren, "Provenance: A Future History," in *OOPSLA*, 2009, pp. 957-964.

[6] G. B. Coe, R. C. Doty, M. D. Allen, and A. Chapman, "Provenance Capture Disparities Highlighted through Datasets," *Theory and Practice of Provenance*, 2014.

[7] H. Conover, R. Ramachandran, B. Beaumont, A. Kulkarni, M. McEniry, K. Regner, and S. Graves, "Introducing Provenance Capture into a Legacy Data System," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, 2013.

[8] P. Groth, S. Miles, and L. Moreau, "PReServ: Provenance Recording for Services," *UK OST e-Science second AHM*, 2005.

[9] R. Laubenbacher, A. Jarrah, H. Mortveit, and S. Ravi, "Agent based modeling, mathematical formalism for," *Computational complexity*, pp. 88-104, 2012.

[10] B. Lerner and E. Boose, "RDataTracker: Collecting Provenance in an Interactive Scripting Environment," *Theory and Practice of Provenance* 2014.

[11] P. Macko and M. Seltzer, "A general-purpose provenance library," *Theory and Practice of Provenance*, 2012.

[12] P. Missier and Z. Chen, "Extracting PROV provenance traces from Wikipedia history pages," *EDBT*, 2013.

[13] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer, "Provenance-Aware Storage Systems," *USENIX*, pp. 43-56, 2006.

[14] T. D. Nies, S. Magliacane, R. Verborgh, S. Coppens, P. Groth, E. Mannens, and R. V. d. Walle, "Git2PROV: Exposing Version Control System Content as W3C PROV," *Proceedings of the 12th International Semantic Web Conference*, 2013.

[15] H. Park, R. Ikeda, and J. Widom, "RAMP: A System for Capturing and Tracing Provenance in MapReduce Workflows," *VLDB*, 2011.

[16] S. F. Railsback and V. Grimm, *Agent-based and individual-based modeling: A practical introduction*. Princeton: Princeton University Press, 2012.

[17] L. Seligman, S. Brady, B. Blaustein, P. Mutchler, A. Chapman, and C. Worrell, "Data Provenance and Financial Systemic Risk," *ICIQ*, 2013.

[18] M. Stamatogiannakis, P. Groth, and H. Bos, "Looking Inside the Black-Box: Capturing Data Provenance Using Dynamic Instrumentation," in *Provenance and Annotation of Data and Processes*, vol. 8628, 2015, pp. 155-167.

[19] U. Wilensky, "Netlogo," *Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.*, vol. http://ccl.northwestern.edu/netlogo, 1999.