Getting It Together: Enabling Multi-organization Provenance Exchange

M. David Allen, Adriane Chapman, Barbara Blaustein, Len Seligman
{dmallen, achapman, bblaustein, seligman}@mitre.org
The MITRE Corporation

Abstract

We present an architecture that supports provenance queries in large, dynamic, multi-organizational environments. The Provenance Challenges have explored *exchange* across disparate provenance systems, yet this is only a first step. We describe requirements for multi-organizational provenance, evaluate candidate architectures, describe the approach implemented in the PLUS prototype provenance manager, and present performance results that indicate the approach is scalable.

1. Introduction

Provenance, the record of origin and manipulations, is a useful tool for helping users understand and use their data appropriately. A small amount of provenance is often better than nothing, but its utility increases as more relationships are tracked, even if tracking many relationships requires user assistance in manipulating the information, such as [1]. However, current system architectures make it difficult to easily extend the amount of provenance available to a user.

Until now, provenance systems have been either a single system keeping track of what a user or group of users did within that individual system [4, 7, 9, 16], or a group of pre-determined heterogeneous systems keeping track of what multiple users did with well-defined processes and data [3, 17]. Current systems are set up to manage provenance within their own walls, which often mirror organizational boundaries; yet, inter-organization information sharing is becoming common. Users need answers to their provenance queries even when provenance graphs cross organizational boundaries.

Enabling provenance exchange amongst disparate provenance systems has been the focus of the Provenance Challenges [12, 15]. Using a standard exchange format [11], individual provenance systems execute a given workflow, exchange provenance in the specified format, and then answer provenance-related questions based on provenance generated from a different provenance system.

However, the ability to exchange provenance information, while an important first step, is hardly sufficient to enable provenance in large-scale multi-organizational environments. When there are many participants with large, dynamic stores, it is impractical to replicate all provenance across all stores to answer any possible user query. We are aware of no prior work that describes a realistic architecture for multi-organizational provenance that meets the following desiderata:

a. **scalability**: There should be a very high limit (or ideally none) to the number of participants. With more participants, the resulting provenance can be richer.

b. **agility**: When new participants join the network, other participants are not required to change software or have knowledge of the new participants. When participants use the same data sets unbeknownst to one another, they can discover each other's provenance without prior knowledge of each other's existence.

c. **autonomy**: Providers retain tight control over their own data, including who they are willing to share with. Without this requirement, fewer organizations will be willing to provide information.

d. **resiliency**: Even if some stores fail, the remaining stores are still available for use.

Figure 1 illustrates some of the challenges. It shows three provenance stores, each containing only part of a potentially huge, multi-organizational graph. Because provenance graphs are inter-linked in ways that are unknown *a priori*, this challenge is a mixture between a data exchange and a discovery problem. Provenance software must discover and connect data sources without knowing ahead of time which data sources or connections need to be made. This differs from more traditional integration problems where only a single step of discovery is necessary, followed by one or more exchanges. Users need answers to their provenance queries as if the full graph were replicated locally. We propose an architecture that meets this need. Our contributions are that we:

1. Enumerate the architectural requirements necessary for true multi-organizational provenance exchange,

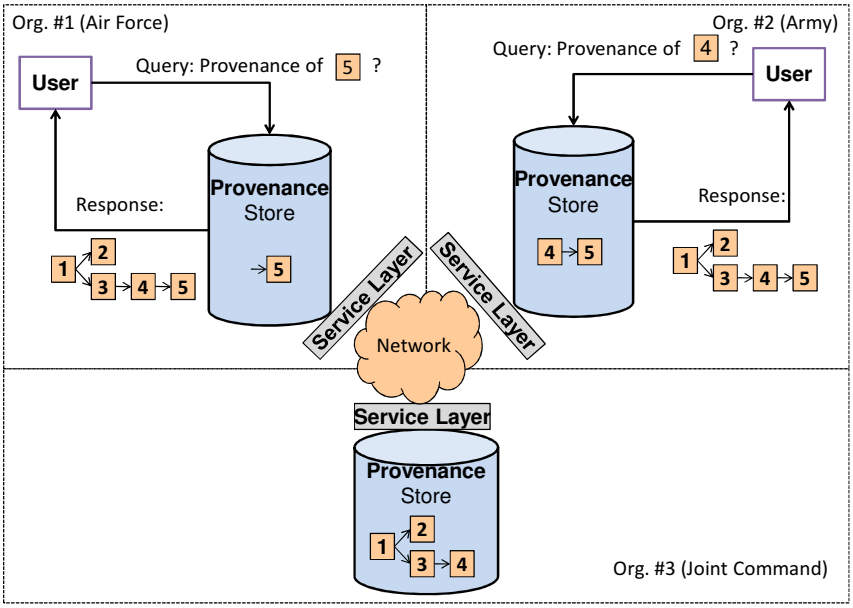

Figure 1: An example of provenance stored across many disparate provenance systems. Users from any node should be able to see query results as if the global graph were stored locally.

2. Analyze several standard architectural paradigms for usage with provenance,
3. Identify a candidate architecture for discovering and exchanging provenance information among many individual provenance systems,
4. Propose a mechanism to identify the same artifact across multiple different systems, independent of the way that the artifact is accessed, and
5. Present an algorithm for assembling a provenance graph from many different provenance systems.

2. Architectural Alternatives

We now consider candidate architectures for multi-organizational provenance. Assembly of provenance graphs poses extra challenges beyond typical data exchange. In a typical data exchange, the sought-after data is specifiable ahead of time, e.g. return all data with "animals" in the title. The exact data and where it resides is not known in advance, but the query for it is self-contained in advance. In the case of multi-organization provenance, the provenance graph is distributed across many different data stores, and the entire set of nodes and edges are not specifiable in advance. Only the set of edges incident to the node of interest are specifiable at any one time. There is no service to connect to that can provide the needed data set; to "connect the dots"; software must repeatedly discover and query many previously unknown stores in order to assemble a larger data structure (the provenance graph) by traversing the currently discovered set of edges. Thus, this data exchange challenge is different from many others in two ways: first, no one entity has the entirety of the desired data and two, recreating the data involves repeated automatic service discovery and query without a human in the loop.

2.1. SOA

Each individual provenance manager can expose the information it manages through a query service. If provenance sharing is desired between organizations, it is possible for one provenance manager to query another via this service interface. Given the current Provenance Challenge methods for provenance exchange, i.e. produce a serialized XML document and send it to another player, this method is a natural first step for multi-organizational provenance exchange. However, this approach limits the extent to which provenance can be discovered and exchanged.

When a system X wants to request information from a provenance service P, X must know about P before it can issue any requests for information. This clearly violates the **agility** property, particularly at large scales, where no single service/UDDI catalog can be assumed. Moreover, if you must know about a provenance service in advance, it constrains the actual number of participants possible, thereby violating the **scalability** property.

2.1. Federated Systems

Another option is a federated system in which a central node provides access to the information in the individual provenance stores. However, this architecture has the following problems:

1. All individual provenance stores must trust the federated store (controlled by some third party) to store or publish their information appropriately, violating the **autonomy** property.
2. As demand for provenance grows, multiple federations will arise; there is typically no central, globally trusted entity to operate a single gateway. Integrating multiple federations then becomes a large problem, clearly violating **scalability**. Even integrating a new provenance store within an existing federation violates **agility**.
3. If the centralized node/federated server is attacked or taken offline, all provenance information is unavailable, violating the **resiliency** property.
4. Someone must fund, maintain and operate the federation. While this does not violate any desiderata, it lowers the probability of this solution being used.

2.2. P2P

A standard P2P arrangement is a federation of equal peers. While there may be intermediate "infrastructure" nodes that aid routing of individual messages to address network topology issues, P2P is **resilient** in that there is no single coordinating authority. P2P networks implement "overlay networks," in essence a virtual network of peers, each connected to one another by nature of their participation in the P2P network. Each peer in the overlay network exposes a certain number of services that can be called by all other peers. In order to join the network, a system must guarantee to expose a certain set of services (as well as meet any other criteria the network may specify), but once it joins, standard P2P routing techniques support discovery over all other peers in the network, satisfying **scalability** and **agility**. Finally, **autonomy** is supported since each individual system only exposes information to the participants it authorizes. There is no need for trust in a third party to store, distribute, route, or inspect any message traffic.

3. Enabling Multi-organizational Exchange

3.1. Artifact Identification

A content-bound artifact identification scheme assigns an identifier as a function of the contents. This is essential because a person who has the data also can identify the data's provenance records without any additional information. Identifiers that are not content-bound may be used, but for the purposes of multi-organizational provenance exchange using multiple provenance managers, content-bound identifiers are essential.

Content-bound identifiers are particularly interesting because they provide traceability: given a data set, we can trace whether the data set has the same identifier as one previously seen. Most content-bound identifiers though, do not work in the reverse; e.g., content hashing represents a 1-way compression operation. Given the data, we can determine the identifier, but we cannot determine the data with only the identifier. Because provenance records what happened to data, the data involved in the provenance never changes, and there is no need to account for potential changes in the content-bound identifier. While old and new versions of a data object are distinct, the provenance information may link them via the process that created the new version.

3.2. P2P Framework Needs

The P2P network maintains a distributed hash table consisting of advertisements published by the peers. These advertisements consist of the unique identifier, and other details, of the data resource being advertised. In the case of provenance information, the unique identifiers refer to provenance nodes (that is, invocations or data) and the additional data provided in the advertisement indicates whether the peer has details about that node, edges incident to that node identifier, or both. Users discover provenance graphs by finding the node that corresponds to the data they possess, and then tracing through the provenance graph.

All sharing architectures require their participants to implement certain contracts so that each peer can rely on the others for an agreed on set of services and interfaces. The basic requirements to make the architecture in Figure 1 effective are:

1. The list of provenance node advertisements that indicate what the peer's provenance holdings are, and
2. A "Provenance Node Access Service". The input is a list of content-bound node identifiers, the credentials of the user, attribute or role that is

```
Function DiscoverProvenance(ID startingPoint):
Queue = (startingPoint);

Results = ({}, {}); Seen = {}

While Queue is not empty:
  ObjID = removeFirst(Queue)
  if seen contains ObjectID: continue;
  Advertisements = getAdvertisements(ObjID)

  For ad in advertisements:
    (Nodes, Edges) = Service(ad.RemotePeerAddr, ad.ObjID)
    Results.addNodes(Nodes);

    for edge in Edges:
      if seen does not contain edge.fromId:
         Queue.add(edge.fromId)
      if seen does not contain edge.toId:
         Queue.add(edge.toId)
      Results.addEdge(edge)
```

Figure 2: Pseudocode for provenance discovery & data exchange in a multi-organizational setting.

requesting the information. The output is a collection of provenance objects matching those identifiers that the provenance store deems releasable given the input credentials.

3.3. Assembling the Complete Graph

Assembling the provenance graph requires: discovery (finding out which store contains which provenance item), requesting the provenance items, and connecting all results into a final provenance graph. Provenance information requires repeated discovery and query, combined with information linkage. This differs from traditional P2P architectures, where the user obtains an entire data item at once. Figure 2 contains pseudo-code which implements the discovery/fetch cycle necessary to retrieve provenance graphs from the P2P network. It contains the following steps:

1. Query advertisements for Node X.
2. Return all advertisements with any information (node, annotation, edge) about X, i.e., find out which peer has which information.
3. Query the peers directly and provide credentials for the provenance information (nodes, annotations, edges).
4. Inspect each response for other provenance node identifiers not yet seen (e.g., an identifier returned as part of an edge description).
5. Assemble the provenance graph, with the results of several iterations of steps 1-4.

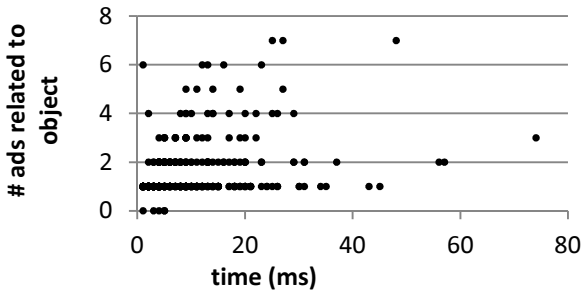

Figure 3: Time to discover provenance objects from multiple provenance managers.

4. Evaluation

4.1. System Implementation

For our proof of concept implementation, we used JXTA[1] version 2.6 for the P2P backbone. Because we were not

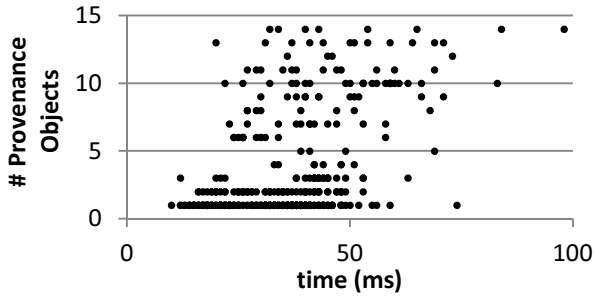

Figure 4: Time to retrieve provenance objects from multiple provenance managers.

[1] http://java.sun.com/othertech/jxta/index.jsp

assessing exchange formats, we used multiple instances of a single provenance system, PLUS [2]. Jena v 2.6.2 generated the data content. Our network consisted of three peers: two were provenance stores, connected as JXTA Rendezvous peers, and the third was a "client" that issued all requests and captured the data presented in the experiments. The client was configured as a JXTA Edge peer.

All experiments were run on a CentOS 5.5 server running Linux kernel 2.6.18 on a single Intel quad core CPU at 2.66Ghz/core with 4GB total RAM, 8GB of swap space, and JVM version 1.6.0_11. In order to minimize network latency, all three peers were run on the same server, communicating via TCP but not using JXTA's option for HTTP exchange.

4.2. Experiments

Figure 3 shows the time to discover provenance objects over multiple managers. Figure 4 shows the time it takes to query those managers and retrieve the requested provenance objects; each dot represents a timed query/response interaction, varying in amount of time necessary (x-axis) and number of results returned (y-axis). In general, query times (Figure 4) are longer than just discovery (Figure 3) since several additional steps occur for a query request. The remote host does the following: parses and interprets; queries the database; converts the results to an RDF/XML string; zips the resulting XML and base 64 encodes it; places it into JXTA resolver query message and sends it. The receiving host decodes, unzips to RDF, and maps the RDF to an internal provenance API. We expect to tune the system to lower costs.

These preliminary experiments show that the P2P Framework can be adapted to handle provenance information, with acceptable graph query and creation time. Although the experiment included only a small number of nodes, the scaling of such networks is well understood, and is unlikely to create substantially different performance figures for larger numbers of peers in the P2P network [18].

5. Related Work

In [5], because the systems in which provenance is being captured are Grid and Web Services, the system focus is SOA. However, no statement is made about the type of architecture that is necessary to the provenance system. However, the setup described in this work is in agreement with the fundamentals described in[5]: a provenance system must support capture, storage, querying and management; query results are created by tracing back edge by edge through the DAG structure and are limited by other filters (such as depth of traversal), etc. In [5], three distinct provenance store setups are described: single, shared, separate. In this work, we agree that each of these setups are valid and necessary to support. However, [5] manages the possibility of storing p-assertions in different provenance stores by locally saving a link to the provenance store in which the p-assertion is stored. Given the requirement that organizations may join and leave the network at will, and that we may wish to query organizations for deeper provenance than we have current knowledge of (by tracing paths in the DAG), we have created a more flexible arrangement.

PReServ [6], with its SOAP Web Services messaging layer, is closest to being ready for distributed provenance queries, but has to this point been used only as a standalone repositories. Other systems such as PASS [13], Panda [8], Taverna [14], Vistrails [16] and ES3 [4] have to this point been used only as a single repository. Even ProvManager [10], which allows execution of multiple workflow systems in a distributed environment (Taverna and Vistrails) still reports into a single provenance repository. The goal of our work is to provide an architecture by which any of these systems can remain as they are, support a very limited service contract, and become a part of a fully distributed provenance system. In other words, we are not aiming to capture provenance in distributed systems, but query over distributed provenance systems.

6. Conclusions and Future Work

Provenance will be most useful when systems can be unaware of its capture, exchange data without sharing a single provenance system, and still create full and complete provenance graphs in response to user queries. Given the constraints of provenance exchange, we have assessed several architectures and determined one which is particularly suitable for provenance exchange. We have performed a basic implementation as a proof of concept, and present initial performance metrics that show that true on-the-fly provenance exchange is possible.

7. Bibliography

[1] O. Biton, S. Davidson, S. Khanna, and S. Roy, "Optimizing user views for workflows," in *ICDT*, 2009, pp. 310-323.

[2] B. T. Blaustein, L. Seligman, M. Morse, M. D. Allen, and A. Rosenthal, "PLUS: Synthesizing privacy, lineage, uncertainty and security," *ICDE Workshops*, pp. 242-245, 2008.

[3] B. Cao, B. Plale, G. Subramanian, P. Missier, C. A. Goble, and Y. L. Simmhan, "Semantically Annotated Provenance in the Life Science Grid," in *SWPM*, 2009.

[4] J. Frew, D. Metzger, and P. Slaughter, "Automatic capture and reconstruction of computational provenance," *Concurr. Comput. : Pract. Exper.*, vol. 20, pp. 485-496, 2008.

[5] P. Groth, S. Jiang, S. Miles, S. Munroe, V. Tan, S. Tsasakou, and L. Moreau, "An Architecture for Provenance Systems," *Technical Report, ECS, University of Southampton*, 2006.

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

[7] D. A. Holland, M. I. Seltzer, U. Braun, and K.-K. Muniswamy-Reddy, "PASSing the provenance challenge," *Concurrency and Computation: Practice and Experience*, vol. 20, pp. 531-540, 2008.

[8] R. Ikeda and J. Widom, "Panda: A System for Provenance and Data," *TaPP*, 2010.

[9] B. Liu, L. Chiticariu, V. Chu, H. V. Jagadish, and F. Reiss, "Refining Information Extraction Rules using Data Provenance," *IEEE Data Eng. Bull.*, vol. 33, pp. 17-24, 2010.

[10] A. Marinho, L. Murta, C. Werner, V. Braganholo, E. Ogasawara, S. M. S. d. Cruz, and M. Mattoso, "Integrating Provenance Data from Distributed Workflow Systems with ProvManager," *IPAW*, 2010.

[11] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. Van den Bussche, "The Open Provenance Model core specification (v1.1)," *Future Generation Computer Systems*, 2010.

[12] L. Moreau, B. Ludäscher, and e. al, "Special Issue: The First Provenance Challenge," *Concurrency and Computation: Practice and Experience*, vol. 20, pp. 409-418, 2008.

[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. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences: Research Articles," *Concurr. Comput. : Pract. Exper.*, vol. 18, pp. 1067-1100, 2006.

[15] ProvenanceChallenge, "http://twiki.gridprovenance.org/bin/view/Challenge/WebHome."

[16] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. Silva, "Querying and Re-Using Workflows with VisTrails," *SIGMOD*, 2008.

[17] W. Tan, R. K. Madduri, A. Nenadic, S. Soiland-Reyes, D. Sulakhe, I. Foster, and C. Goble, "caGrid Workflow Toolkit: A Taverna based workflow tool for cancer Grid," *BMC Bioinformatics* vol. 11, 2010.

[18] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 41-53, 2004.