acmqueue

Provenance in Sensor Data Management

A cohesive, independent solution for bringing provenance to scientific research

Zachary Hensley, Tennessee Technological University

Jibonananda Sanyal, Oak Ridge National Laboratory

Joshua New, Oak Ridge National Laboratory

In today's information-driven workplaces, data is constantly being moved around and undergoing transformation. The typical business-as-usual approach is to use e-mail attachments, shared network locations, databases, and more recently, the cloud. More often than not, there are multiple versions of the data sitting in different locations, and users of this data are confounded by the lack of metadata describing its provenance—or in other words, its lineage. The ProvDMS project at the Oak Ridge National Laboratory (ORNL) described in this article aims to solve this issue in the context of sensor data.

ORNL's Building Technologies Research and Integration Center has reconfigurable commercial buildings deployed on FRPs (flexible research platforms). Figure 1 is a Google Earth model of a medium-size commercial office building that is part of the ORNL's FRP apparatus. These buildings (metal warehouse and office) are instrumented with a large number of sensors that measure variables such as HVAC efficiency, relative humidity, and temperature gradients across doors, windows, and walls. The sensors acquire sub-minute resolution data from hundreds of channels. Scientists conduct experiments, run simulations, and analyze the data. The sensor data is also used in elaborate quality assurance exercises to study the effect of systemic faults. The two types of commercial buildings

FIGURE
1

Model of a Building that is Part of the ORNL's FRP Apparatus

comprising the FRPs stream data at a 30-second resolution for a total of 1,071 channels for both buildings.

The sensor data collected from the FRPs is saved to a shared network location accessible by researchers. It became apparent that proper scientific controls required not just managing the data acquisition and delivery, but also managing the metadata associated with temporal subsets of the sensor data. The ProvDMS, or Provenance Data Management System, for the FRPs allows researchers to retrieve data of interest, and to trace its lineage. The life cycle of most objects consists of creation, curation, transformation, archival, and potentially deletion. *Provenance* is the tracking of such information.⁸

ProvDMS provides researchers with a one-stop shop for all data transformations, allowing them to effectively trace their data to its source so that experiments and derivations of experiments can be reused and reproduced without the overhead of repeating every experiment.

RELATED WORK

There are a number of existing software systems for provenance data collection with strong work-flow integration. Chimera⁶ is a process-oriented provenance system that manages derivation and analysis of data objects in collaboratory environments. It stores provenance information that can be used to regenerate, compare, and audit data derivations within the system. The Karma provenance system⁷ allows users to collect and query the provenance of scientific data processes with the ability to run either stand-alone or as part of a larger cyber-infrastructure setup. The Karma system is intimately connected with its data as a result of its close work-flow integration. VisTrails^{4,5} provides support for scientific data exploration and visualization with a strong focus on work flow as provenance objects to represent complex computations. Work flow in VisTrails can be visualized as pipelines of procedure sequences that lead to a computational output. The EU Provenance Project¹ uses an open provenance architecture for grid systems with a service-oriented approach, and has been used for aerospace engineering and organ-transplant management. The provenance system was used to track medical information in units of patient-doctor interactions. The project attempted to find equilibrium between the amount of data collected and minimizing the intrusiveness of the collection effort in order to preserve the quality of medical care.

While many of these systems are complete software solutions, CPL (Core Provenance Library)³ was designed to be application-independent and easy to integrate into new or existing systems. Because of its independent nature, CPL was used in ProvDMS as the provenance back end. This allowed the user interface of ProvDMS to be separate from CPL's object constraints, thereby producing a positive user experience.

EFFECTIVE PROVENANCE DESIGN

Particular implementations of provenance can vary greatly depending upon a few important attributes. The focus of ProvDMS is on researcher requirements, granularity of the provenance, work-flow requirements, and object design. Its design principles emphasize the importance of user needs, taking a cohesive but independent stance on the integration of provenance with user tools.

- **Granularity.** Most systems incorporate both fine and coarse granularity to avoid restricting the type and amount of data available to users.² ProvDMS implements a fine-granularity system but provides a mixed-granularity interface for users so that tracing lineage using visualization is

contextual. Users are shown generalizations as coarse provenance objects that can be contextually expanded to provide finer-granularity information. This allows users to see still finer, exact provenance objects that specifically map to logical objects in the system, but users are not overburdened with unnecessary information when viewing the provenance.

- **Tool integration with work flow.** ProvDMS's design was largely determined by the “when” and “how” of integrating with existing tools. Most tools have limited or no provenance-tracking abilities. ORNL researchers routinely use a wide array of specialized tools from various vendors that do not have provenance support. As a result, ProvDMS could not have many restrictions. While challenging, this steered the focus toward data infrastructure requirements to enable tracing the provenance while facilitating development of software interfaces to support future system integration. To enable a sense of work flow, ProvDMS uses the notion of a user *experiment*, where the sensor data resides once exported from the system. Users may choose any tool and have the option of importing different states of their experiments back into ProvDMS.

- **Provenance of provenance.** The ability of a provenance system to track how it creates and tracks provenance objects was not an initial design requirement for ProvDMS but emerged from the abilities of CPL. By tracking PoP (provenance of provenance), ProvDMS provides specific information about when the provenance system created new objects or versions of objects, which user was responsible for the creation of objects, the process ID that performed tracking functions, and system information such as the executing environment. Administrators of this system can now track system usage over time and may detect patterns in how the system and provenance data storage is being used.

- **Uniqueness.** Provenance systems inherently involve hierarchical connectivity among objects. The use of CPL as the provenance back end allows users to access provenance object ancestry easily. Additionally, CPL's versioning system ensures that each object is uniquely identifiable, which solves the design issue of the user's ability to define an experiment multiple times.

- **Object design.** Object design shaped the entirety of ProvDMS and arguably comprised the most difficult set of decisions to make. The first challenge was to determine how users are expected to interact with the data that would determine the required provenance objects. This was difficult to gauge for a system that was still on paper. We were unsure of the level of granularity of provenance to store and expose since it was possible that much of the provenance data could go unused. We leaned toward finer granularity while providing support across a spectrum of granularity to account for the yet-to-be-discovered unknowns in ProvDMS.

Provenance objects in CPL are uniquely defined using three main attributes: *Name*, *Type*, and *Originator*. In ProvDMS's use of CPL, *Name* describes the object, and *Type* determines its granularity. The *Originator* is intended to be used in a similar vein to Java's package-naming convention, via hierarchical domain namespaces. ProvDMS uses the name of the system as the top-level domain, user as the next level, and interface as the final domain. This ensures the existence of understandable and unique originators differentiating the *experiments* (and corresponding provenance objects) based on authenticated users.

SYSTEM ARCHITECTURE, DESIGN, COHESION, AND INDEPENDENCE

The FRPs use Campbell Scientific's data loggers to collect data from 1,071 channels in the facility. Campbell Scientific's LNDB (Loggernet Database) runs on a dedicated server and populates a MySQL

FIGURE 2

Logical Representation of the Physical Layout of FRP Data

database with the raw sensor output. ProvDMS runs on another dedicated server and retrieves the data from the MySQL database to fulfill user needs, thereby providing complete separation of the raw data store from the provenance trace. LNDB creates the required schema on the data server, and ProvDMS is architected to sense the schema and its logical relationship to the FRP in order to present a cogent, simplified interface to the users. Checks are in place to ensure data backup, security, and isolation since much of the data is proprietary.

Figure 2 shows the logical representation of the physical layout of FRP data. This influences the provenance object design of ProvDMS. As illustrated in the figure, the sensor data is separated into *stations*, each containing a set of *data loggers*. These data loggers consist of sets of data *channels*. Physically, these channels relate to *sensors* placed in different locations throughout the test facility.

The ultimate goal of the provenance system is to trace the participation of temporal subsets of sensor data in user experiments. ProvDMS treats these as objects. Researchers export a temporal subset of the chosen sensor channels as an experiment, which can then go through various transformations in the user's workspace. Once researchers feel ready, they may submit the "state" of their experiment to the system, along with any additional derived data, supporting files, results, or other metadata. ProvDMS allows users to map the uploaded files as a derivative of the original experiment.

The logical representation of FRP data was designed to correspond to the provenance objects. Each *type* of provenance object relates to its logical representation. These objects have similar representations, with a few differences. Importantly, there are additional links from data loggers to their associated files. In the case of user-defined experiments, these are the files holding channels of sensor data. For derived experiments, these are any associated files used in the derivation. Figure 3 illustrates the differences in their representations. Two types of links are used: version dependencies and data-flow dependencies. Differences between these links are important for the cycle-avoidance algorithm in the CPL.

FIGURE 3

Logical Representation of the Provenance Object Design

FIGURE 4

The Layers and Components of ProvDMS

In addition, there are two types of links between objects: version dependencies are used for objects that are created as new versions of previous objects; and data-flow dependencies are used as ancestry links between differing objects, representing a translation of data between them. The differences between these two types of links are very important for CPL's cycle-avoidance algorithm (discussed in more detail later in the section covering provenance visualization).

Architecturally, ProvDMS has a layered design. Figure 4 is a diagram showing its layers and components. The compatibility layer includes two wrappers: a PHP wrapper and a C++ wrapper that the PHP wrapper interacts with. The C++ wrapper abstracts the provenance back-end interaction. The different components interact cohesively:

FIGURE 5

The Experiment Creation Interface

- The ProvDMS layer represents the user interface for provenance interaction. Figure 5 shows the experiment creation interface, which allows users to select subsets of data for experiment objects and to interact with managed sensor data, visualize provenance information, and either define or derive experiments.
- The compatibility layer abstracts the API calls of CPL so ProvDMS can interact with the underlying system. Using the PHP wrapper, the system can pass queries from the coupled software to the database back end, as well as share those results.
- The provenance database is the storage layer of ProvDMS. The interactions between the database and the compatibility layer allow for provenance information to be gathered when users define or derive experiment objects while interacting with ProvDMS.
- The sensor database stores FRP sensor data, as well as stored procedures for fast querying when needed. Most data retrieved from this layer is joined with particular FRP stations or data loggers before transmission. It is independent of the provenance system in order to facilitate decoupled scalability and interfacing with other software components being developed for the FRPs.

Using CPL allows ProvDMS to act independently of provenance-calling API hooks when information has to be saved to the provenance database. An abstraction layer handles the translation of user actions to CPL API calls for inserting or querying provenance information. This is encapsulated into a compatibility layer containing the PHP and C++ wrappers.

The PHP code in figure 6 demonstrates the wrapper's interaction with C++ in order to store or retrieve provenance data.

CPL, which is written in C, already includes some C++ functionality. The C++ wrapper abstracts the interaction with CPL via a heavily object-oriented interface. The code snippet in figure 7 illustrates the creation of provenance objects. As illustrated, PHP communicates with the C++ wrapper using `exec` calls. The decision to forgo a PHP extension was based on a few driving factors:

- **Trade-off.** The trade-off between decoupled generality and the performance overhead of `exec`

FIGURE 6

The Wrapper's Interaction with C++

```
function prov_new_experiment( $experiment )
{
    global $command;
    global $userID;

    $retInfo = Array();
    $retStatus = null;

    $exp = new Provenance_Object( $experiment['name' ], "Experiment" );
    $exp->addProperty( 'time_begin', $experiment['time_begin'] );
    $exp->addProperty( 'time_end', $experiment['time_end'] );

    $params = '';
    $params .= ' -c "create_object"';
    $params .= ' -n "' . $exp->getName() . '"';
    $params .= ' -t "' . $exp->getType() . '"';
    $params .= ' -o "' . $exp->getOriginator() . '"';
    $params .= ' -p "' . $exp->getProperties() . '"';
    $params .= ' -u "' . $userID . '"';
    $params .= ' -s "Yes"';

    //Soft-create, make new version if already exists

    exec( $command . $params, $retInfo, $retStatus );
}
```

calls, especially for a small number of them, leaned toward a PHP exec framework rather than a full PHP extension.

- **Simplicity.** Using an exec call to an external C++ executable made it possible to maintain a simple parameter-based call similar to that of bash.
- **Source.** Including the C++ wrapper as an external executable while providing source code allows administrators to modify the wrapper based on organizational needs.
- **Time to implement.** ProvDMS was designed and implemented in eight to nine weeks, and we made the best of the rapid development. A complete PHP extension implementation was outside the scope of the allocated time and budget for the project.

The integration with CPL was among the smoothest parts of ProvDMS's implementation. Some minor differences in testing and using CPL-integrated systems on different client and server platforms exist. Open-SUSE 12.3 was used for development and testing of ProvDMS, and Red Hat

FIGURE 7

The Creation of Provenance Objects

```
bool hook_create_object( const char * user, Prov_Params params )
{
    int pid = getpid();

    odbcHandler * handler = new odbcHandler( "CPL", true, user, pid );
    handler->new_object( params );
    delete handler;

    return true;
}
```

Enterprise Linux 6 for the production version.

One of the earliest hitches involved interactions between PHP and `exec'd` C++ calls. In order for CPL to provide PoP, it must pull some information from the executing environment. This works perfectly for client-side execution of CPL code, but once the CPL code is executed via PHP `exec` calls, certain environment variables are no longer retrievable. These variables are necessary to save information about provenance sessions, and thus the provenance back end cannot continue. A quick hot-fix to pass in proper environment information evaded the pull from environment variables.

FEATURES AND USAGE

ProvDMS was built not just to trace the provenance of experiments, but also to be a one-stop access point for all sensor data-related activities for the FRPs. It provides the following interface features:

- **Experiment creation.** ProvDMS allows users to select subsets of stations, data loggers, and channels as a definition of a new experiment. This information is parsed and saved as CSV files on the server. On request, users can export this data. On creation, each experiment is defined as a provenance object by the provenance back end, which also creates all finer-granularity objects.
- **Experiment derivation.** Users upload and define experiments as derivations of previous experiments. This means users can save the state of their data and any associated files in ProvDMS, allowing them to trace the derivation in the future.
- **Data status.** The system provides a dashboard with Sparklines,⁹ which helps summarize the status of data on the server. Sparklines are small trending plots that have no axis labels, allowing them to be embedded inline with text, thus permitting users to pick out trends in the data easily. Sparklines can display the status of key channels from different sensors for quick assessment and detection of faults in the system.
- **Provenance visualization.** The system provides visualization capabilities so users can easily visualize their data's lineage. The subject of provenance visualization warrants a separate discussion, and is covered in more detail in the next section.

Much of the early development of ProvDMS was spent ensuring that it is natural and simple

FIGURE 8

Logical Representation of a Subset of Provenance Data

to use. For example, the experiment creation feature (figure 5) is designed with effective user interaction principles to enable a simple “flow,” and it emphasizes the importance of efficiency when managing user data.

PROVENANCE VISUALIZATION

The first question anyone should ask when beginning visualization is similar to the first question that should be asked when designing a provenance system: “What information is important?”

The developers of CPL suggest the use of their Orbiter tool to visualize provenance using CPL. Orbiter is an external visualization program developed in Java. It pulls information from the CPL database (an SQL back end in this case) and visualizes it using a node-link graph. It includes features for time-based visualization and node grouping for nodes with common links. It is an excellent tool for visualizing the information from the CPL database.

As easy as it would have been to make Orbiter ProvDMS’s visualization tool, there were some issues. Primary among these is CPL’s use of a cycle-avoidance algorithm to version and link objects without creating cycles in object provenance. Contextual information must be displayed as part of the visualization. This means removing particular information from CPL’s ancestry queries. Figure 8 shows a subset of provenance information created by ProvDMS and its integration with CPL. Two versions of the finer-granularity objects exist as a result of data-flow dependencies and the cycle-avoidance algorithm. These extra nodes must be removed for clean visualization of the provenance. This information is correct in its representation, but many of the objects are not important to users and can obfuscate the data’s lineage in the visualization. For a clearer representation of the provenance, the double versions created via the translated objects as a result of cycle avoidance must be removed.

It is important to note how specific these parsed cases are. In the figure, the experiment objects

FIGURE 9

The Force-Based Node-Link Layout

are missing the extra translation versions because these experiments are linked only via version dependencies. This means a user has created a new experiment with the same identification as a previous experiment. This is a cue for ProvdMS's integration with CPL to create a new version of this experiment. This procedure bypasses the need to link objects manually via data-flow dependencies. A situation like this increases the difficulty in parsing individual cases for visualization.

TYPES OF VISUALIZATION

ProvdMS's visualization is Web-enabled using various JavaScript visualization libraries such as the JIT (JavaScript InfoVis Toolkit) and D3.js (Data Driven Documents). ProvdMS includes a few types of visualizations:

- **Non-unique node-link tree.** Objects in CPL's provenance implementation are inherently unique because of the Name, Type, Originator object convention. Though objects are initially created uniquely, the nature of provenance is to provide a hierarchical flow of data. Objects will undoubtedly have multiple versions at some point in their life cycles. Multiversed objects do not change their identification from one version to another. As only their version changes, the nodes are no longer

FIGURE 10

A Close-Up of One of the Groupings in the Forced-Based Node-Link Layout

uniquely identified using the same convention for this type of visualization.

• **Force-based node-link layout.** Classical approaches to the visualization of provenance focus on tree-like views rooted from the top-level provenance object (often a selected one). CPL's objects are designed to use this type of inheritance as well, relying on descendants and ancestors for the traversal of provenance information. Even then, it can be useful to visualize the lineage of objects differently, such as employing a force-based layout. This layout still uses a node-link format—as the other layouts do—but it uses a system of forces acting on each node to determine their positions. This makes the system feel more interactive, as users can apply forces to nodes in the graphs by dragging them.

Figures 9 and 10 demonstrate some interesting results of this type of visualization. In figure 9, a traceable flow of data lineage is visible, as well as a natural grouping of objects with similar granularity. Solid gray lines represent hierarchical connections between provenance objects that

FIGURE 11

ProvDMS's Current Visualization, using Contextual Node-link Trees

group together as information relevant to a single version of a user-defined or imported experiment. Orange nodes represent the top-level experiment objects that are parents of all associated finer-granularity objects such as stations, data loggers, and channels (colored blue, red, and green, respectively). In figure 10, the innermost node and all of the finer-granularity nodes' connections create a pseudo “weight” that encompasses the entirety of the grouping of objects. Each object has its associated weight, which affects the layout of all connected nodes. The grouping tends to act as a single node in the visualization.

- **Unique, contextual node-link tree.** The current implementation of visualization in ProvDMS uses this approach in its provenance-visualization module. Like the first approach, this one uses a node-link tree to visualize the provenance in a hierarchical fashion. Nodes are expanded asynchronously, pulling information from the provenance database as they expand. Contextual information can be shown for certain objects. In this manner, even finer granularity can be visualized by processing provenance object properties in addition to the objects themselves. Figure 11 is an example of this visualization, using contextual node-link trees. Two nodes are expanded to show meta-information at a finer granularity level than their parent nodes. Experiment nodes are the coarsest objects, while information specific to provenance objects, shown in rectangles, is at the finest granularity level.

CONCLUSION

This attempt to bring provenance to scientific research has highlighted some of the challenges and potential solutions for applying provenance to generalized data streams. Although we successfully built a system to handle provenance for ORNL's FRPs, this specific use case makes it less general

than many other provenance systems. The availability of CPL as a library has been beneficial. The success with using CPL can be attributed to ProvDMS being independent of the provenance back end, providing the required flexibility in system design. The C++ and PHP-wrapper code developed during the project was contributed back to the authors of CPL for future integration.

Research efforts are under way in automated sensor-data validation, estimation for missing or corrupt data, and machine-learning estimations of sensor health with plans to integrate work flows with ProvDMS. The systems will connect to the underlying layers of ProvDMS, allowing integrated tracking of the provenance for data validity, fault detection, and quality assurance.

Despite challenging design decisions, usability both guided and restricted the abilities of ProvDMS. We limited the features and the granularity of collected provenance to ensure minimal restrictions and minimize the additional training required of the researchers. The result is a simple interface for users to keep track of their data and experiments manually. The modular design of ProvDMS will allow the addition of newer provenance-collection methods as the system evolves. The knowledge derived from our experience with ProvDMS's design and use should soon lead to further improvements.

In the end, ProvDMS can serve as an example of implementing and using provenance of a common data archetype in an environment normally devoid of information-tracking methods. ProvDMS should demonstrate the power of such systems for enabling reproducible science.

ACKNOWLEDGMENTS

The authors thank Peter Macko and Margo Seltzer of Harvard University, the authors of CPL, for their support and guidance on the use of CPL during the project. This work was funded by fieldwork proposal RAEB006 under the Department of Energy Building Technology Activity Number EB3603000. We would like to thank Edward Vineyard for his support and review of this project.

Oak Ridge National Laboratory is managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725. This manuscript has been authored by UT-Battelle, LLC, under Contract Number DEAC05-00OR22725 with the U.S. Department of Energy. The U.S. government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. government purposes.

REFERENCES

1. Alvarez-Napagao, S., Vazquez-Salceda, J., Kifor, T., Varga, L. Z., Willmott, S. 2006. Applying provenance in distributed organ transplant management. In *Provenance and Annotation of Data*: 28-36. Springer.
2. Chapman, A., Jagadish, H. V. Issues in building practical provenance systems. 2007. *IEEE Data Engineering Bulletin* 30(4): 38-43.
3. Macko, P., Seltzer, M. 2012. A general-purpose provenance library. In the 4th Usenix Workshop on the Theory and Practice of Provenance.
4. Scheidegger, C., Koop, D., Santos, E., Vo, H., Callahan, S., Freire, J., Silva, C. 2008. Tackling the provenance challenge one layer at a time. *Concurrency and Computation: Practice and Experience* 20(5): 473-483.
5. Silva, C. T., Freire, J., Callahan, S. P. 2007. Provenance for visualizations: reproducibility and

- beyond. *Computing in Science & Engineering* 9(5): 82-89.
6. Simmhan, Y. L., Plale, B., Gannon, D. 2005. A survey of data provenance in e-science. *ACM Sigmod Record* 34(3): 31-36.
 7. Simmhan, Y. L., Plale, B., Gannon, D. 2008. Query capabilities of the Karma provenance framework. *Concurrency and Computation: Practice and Experience* 20(5): 441-451.
 8. Szomszor, M., Moreau, L. 2003. Recording and reasoning over data provenance in Web and grid services. In *On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*: 603-620. Springer.
 9. Tufte, E. 2004. Sparklines: theory and practice; http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0001OR&topic_id=1.

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

ZACHARY HENSLEY is a senior computer science major with a focus on software and scientific applications at Tennessee Technological University. He has spent the past two summers as an intern working for Oak Ridge National Laboratory, first with the Neutron Data Analysis and Visualization Group and second with the Energy and Transportation Sciences Division. The ProvDMS project was his first experience of extensive research into data provenance, which has further piqued his interest in applied research for practical applications.

JIBONANANDA (JIBO) SANYAL is a staff scientist at the Building Technologies Research and Integration Center at Oak Ridge National Laboratory. His current research focus includes massively parallel and scalable energy simulations on high-performance computing systems and subsequent big-data analysis. He holds a Ph.D. and an M.S. degree in computer science from Mississippi State University and two B.S. degrees from the University of Calcutta and the North Eastern Hill University, India. His prior research focused on developing and evaluating uncertainty visualization techniques and tools for the fields of meteorology and hydrology.

JOSHUA NEW has been a computer scientist with Oak Ridge National Laboratory since 2009. He coordinates the energy-modeling efforts of ORNL's Building Technology Research Integration Center for EnergyPlus, OpenStudio, and advanced simulation capabilities leveraging high-performance computing and artificial intelligence for building energy models via ensemble simulations and big data mining. He received his Ph.D. in computer science from the University of Tennessee in 2009. He has an M.S. in computer systems and software design and a B.S. with a double major in computer science and mathematics including a physics minor from Jacksonville State University.

© 2013 ACM 1542-7730/13/1200 \$10.00