Provenance In Sensor Data Management: A Cohesive, Independent Solution

Zachary Hensley, Tennessee Technological University
Jibonananda Sanyal, Oak Ridge National Laboratory
Joshua New, Oak Ridge National Laboratory

1. INTRODUCTION

In today's information-driven workplaces, data is constantly undergoing transformations and being moved around. The typical business-as-usual approach is to use email 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. Our project is aimed to solve this issue in the context of sensor data. The Oak Ridge National Laboratory's Building Technologies Research and Integration Center has reconfigurable commercial buildings deployed on Flexible Research Platforms (FRPs) as seen in figure 1. These buildings (metal warehouse and medium office) are instrumented with a large number of sensors which measure a number of variables such as HVAC efficiency, relative humidity, and temperature gradients across doors, windows, and walls. Sub-minute resolution data from hundreds of channels is acquired. A number of scientists conduct experiments, run simulations, and performing analysis with the data. The sensor data also participates in elaborate quality assurance exercises to study the effect of systemic faults. The two commercial buildings 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 are saved to a shared network location which is accessed by researchers for their needs. It became apparent that proper scientific controls required not just managing the data acquisition and delivery, but to also managing the metadata associated with temporal subsets of the sensor data. We built a system named ProvDMS, or Provenance Data Management System for the FRPs, which allows researchers to retrieve data of interest as well as trace its lineage. Provenance is the inherent *lineage* of objects as they evolve over time. The life-cycle of most objects consists of creation, curation, transformation, archival, and potentially deletion. *Provenance* is the tracking of such information [8].

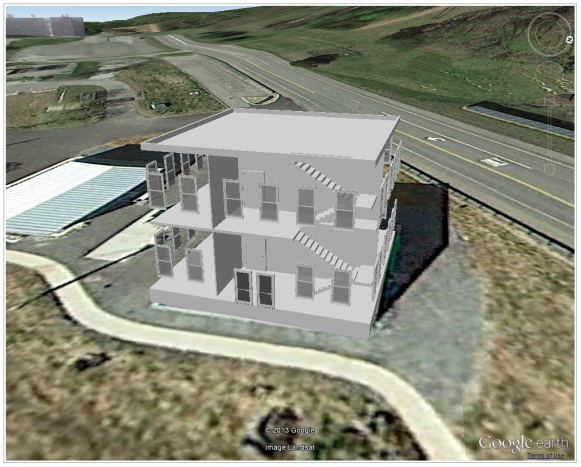

Figure 1: Illustration of a Google Earth model of a medium-office commercial building which is part of the ORNL's Flexible Research Platforms apparatus.

ProvDMS provides researchers 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 be reused and reproduced without the overhead of repeatability of experiments that use it.

2. RELATED WORK

There are a number of existing software systems for provenance data collection with strong workflow integration. Chimera [6] is a process-oriented provenance system which manages derivation and analysis of data objects in collaboratory environments. It stores provenance information which can be used to regenerate, compare, and audit data derivations within the system. The Karma provenance system [7] allows users to collect and query provenance of scientific data processes with the ability to either run stand-alone or as part of a greater cyber-infrastructure setup. The Karma system is intimately connected with its data as a result of its close workflow integration. Vistrails [4, 5] provides support for scientific data exploration and visualization with a strong focus on workflows as provenance objects to represent complex computations. Workflows in Vistrails can be visualized as pipelines of procedure sequences that lead to a computational output.

The EU-Provenance Project [1] uses an open provenance

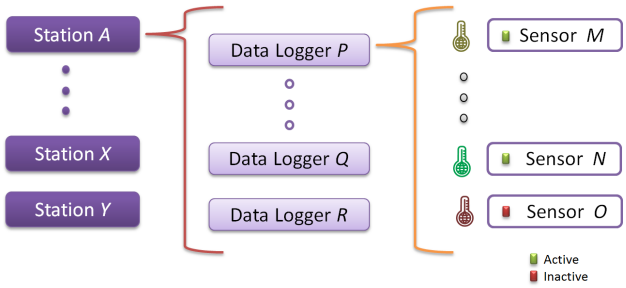

Figure 2: Logical representation of the physical layout of FRP data. This influences the provenance object design of ProvDMS.

architecture for grid systems using a service-oriented approach, namely 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 an equilibrium between the amount of data collected and the minimization of intrusiveness of the collection effort in order to preserve the quality of medical care.

While many of these systems are complete software solutions, the Core Provenance Library (CPL) [3] was designed as a provenance library purposed to be application-independent and easy to integrate into new or existing systems. Because of the independent nature of CPL, we used the library to serve as the provenance backend in ProvDMS. This allowed us to keep the user interface of ProvDMS separate from CPL's object constraints for a positive user experience.

3. PROVDMS – EFFECTIVE PROVENANCE DESIGN

Particular implementations of provenance can vary greatly depending upon a few important attributes. In building our provenance system, we focused heavily on researcher requirements, granularity of the provenance, workflow requirements, and object design. Our 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]. We implement a fine-granularity system but provide a mixed-granularity interface for our users so that tracing the lineage using the visualization is contextual. Users are shown generalizations as coarse provenance objects that can be contextually expanded to provider finer granularity information. This allows users to still see finer, exact provenance objects that specifically map to logical objects in the system but are not overburdened with unnecessary information when viewing the provenance.

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

- *Provenance of Provenance.* The ability for a provenance system to track how it creates and tracks provenance objects was not an initial design requirement for our system but emerged from the abilities of the CPL library incorporated into ProvDMS. By having the ability to track Provenance of Provenance (or PoP), our system 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 like the executing environment. Administrators of our system can now track system usage over time and may detect patterns in system usage and how provenance data storage is being used.

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

- *Object Design.* Arguably the most difficult set of decisions to make, object design shaped the entirety of our provenance system. Our first challenge was to determine how users are expected to interact with the data which would determine the required provenance objects. This was difficult to guage for a system that was still on paper. We were unsure of the level of granularity of provenance to store and expose since there was the possibility that much of our provenance data could go unused. We leaned on the side of finer granularity while supporting across a spectrum of granularity to account for the yet-to-discover unknowns in our system.

  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 the granularity of the object. The *Originator* is intended to be used in a similar vein to Java's package naming convention — via hierarchical domain namespaces. In our system, we use the name of the system as the top-level domain, user as the next level, and interface as the final. This ensures we have understandable and unique originators differentiating the "Experiments" (and corresponding provenance objects) based on authenticated users.

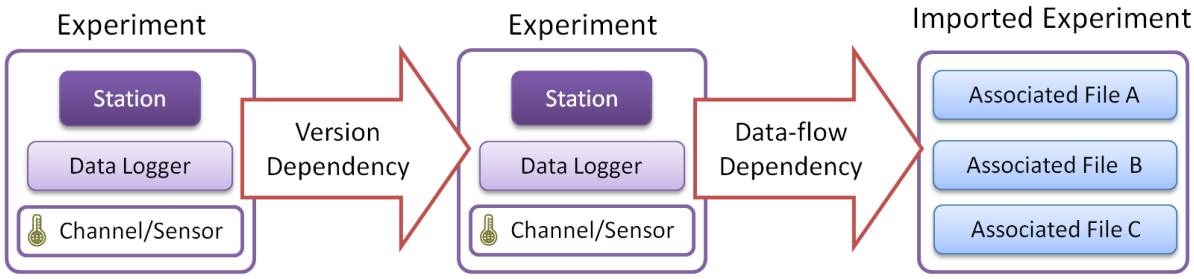

Figure 3: Logical representation of the provenance object design. Two types of links are used: version dependencies and data flow dependencies. Differences between these links are important for the Cycle Avoidance algorithm in the Core Provenance Library.

4. SYSTEM ARCHITECTURE, DESIGN, COHESION, AND INDEPENDENCE

Campbell Scientific's data loggers are used for collecting data from the 1,071 channels in the FRPs. Campbell Scientific's Loggernet Database (LNDB) tool runs on a dedicated server and populates a MySQL database with the raw sensor output. Checks are in place to ensure data backup, security, and isolation since much of the data is proprietary. The ProvDMS system 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. As illustrated in figure 2, the sensor data is separated into *Stations*, each containing a set of *Data Loggers*. These *Data Loggers* consist of a set 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". We treat these as objects in ProvDMS. 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. The ProvDMS system allows users to map the uploaded files as a derivative of the original "Experiment".

We designed the logical representation of FRP data to correspond to the provenance objects. Each *Type* of provenance object relates to its logical representation. These objects are similar in their representation, 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 representation.

In addition, there are two types of links between objects. Version dependencies are used for objects which are created as a new version of a previous object. 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, which will be discussed in more detail in section 6.

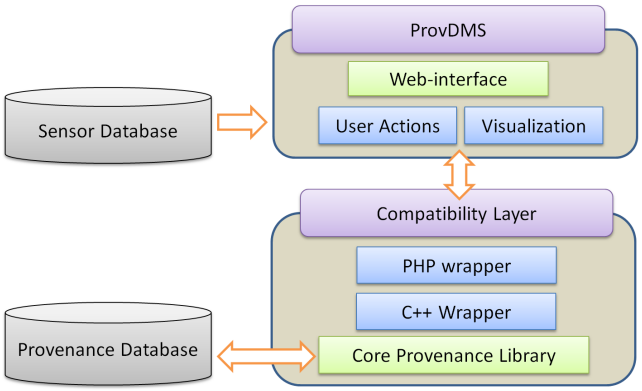

Figure 4: A diagram showing the layers and components of ProvDMS. The Compatibility Layer includes two wrappers: a PHP wrapper and a C++ wrapper that the PHP wrapper interacts with. The C++ wrapper abstracts the provenance back-end interaction.

Architecturally, ProvDMS has a layered design (figure 4) and the different components interact cohesively:

- The ProvDMS layer represents our user interface (figure 5) for provenance interaction. It allows users to interact with managed sensor data, visualize provenance information, and either define or derive experiments.
- The Compatibility Layer abstracts the API calls of CPL to allow ProvDMS to interact with the underlying system. Using the PHP Wrapper, the system can pass queries from the coupled software to the database backend 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 de-coupled 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. To interact with CPL, we built an abstraction layer to handle 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 following PHP code demonstrates the wrapper's interaction with C++ in order to store or retrieve provenance data.

```
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 );

}
```

CPL, written in C, already includes some C++ functionality. Our C++ wrapper abstracts the interaction with CPL via a heavily object-oriented interface. The code snippet below illustrates 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;
}
```

As illustrated, PHP communicates with the C++ wrapper using *exec* calls. Our decision to forgo a PHP extension was based on a few driving factors:

- *Trade-off*: The trade-off between decoupled generality and performance overhead of *exec* calls, especially for a small number of them, predisposed us toward a PHP *exec* framework rather than a full PHP extension.
- *Simplicity*: By using an *exec* call to an external C++ executable, we are able 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*: We have designed and implemented the system in a span of 8-9 weeks. We made the best of rapid development given our short project time. 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. We have successfully used OpenSUSE 12.3 for development and testing of ProvDMS, and use Red Hat Enterprise Linux 6 for the production version.

One of the earliest hitches we encountered involved interactions between PHP and *exec*'d C++ calls. In order for CPL to provide Provenance of Provenance (PoP), it must pull some information from the executing environment. This works perfectly for client-side execution of CPL code. However, 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 can no longer continue. A quick hot-fix to pass in proper environment information ourselves evaded the pull from environment variables.

5. FEATURES AND USAGE

We built ProvDMS to not just trace the provenance of experiments, but be a one-stop access point for all sensor data related activities for the Flexible Research Platforms. The following interface features are provided:

- *Experiment Creation*: We provide users with the ability 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, this data can be exported by users. On creation, each *Experiment* is defined as a provenance object by the provenance back-end – creating all finer granularity objects in addition.

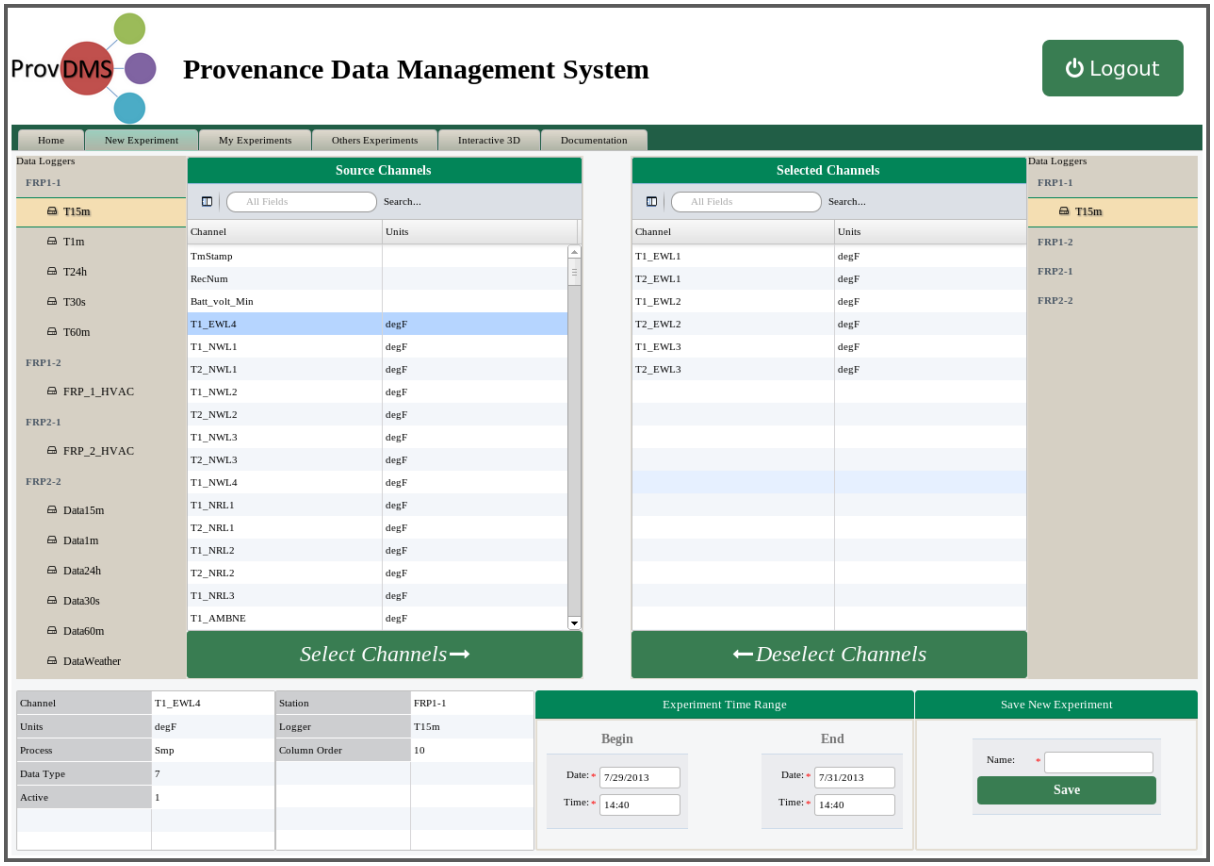

Figure 5: The *Experiment Creation* interface. This interface allows users to select subsets of data for *Experiment* objects.

- *Experiment Derivation*: Users upload and define *Experiments* as derivations of previous *Experiments*. This allows users to 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 to summarize the status of data on the server. Sparklines are small trending plots that have no axes labels which allows them to be embedded inline with text, allowing users to pick out trends in the data easily. We make use of Sparklines to 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 to allow users to easily visualize their data's lineage. The subject of provenance visualization warrants a separate discussion, and we talk more of our attempts to provide adequate visualization in the next section.

We spent much of the early development stages to ensure usage of our system is natural and simple. For example, the *Experiment Creation* (figure 5) feature is designed with effective user interaction principles to enable a simple "flow" and emphasizes the importance of efficiency when managing user data.

6. PROVENANCE VISUALIZATION

The first question anyone should ask themselves when beginning visualization is unsurprisingly similar to the first question they should ask 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 our 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 to visualize the information from the CPL database.

As easy as it would have been to tie in Orbiter as ProvDMS's visualization tool, there are some issues. Primarily among these involves CPL's use of a Cycle Avoidance algorithm to version and link objects without creating cycles in object provenance. We are required to display contextual information as part of our visualization. This means we have to remove particular information from CPL's ancestry queries. Figure 6 shows a subset of provenance information — created by ProvDMS and its integration with CPL. This in-

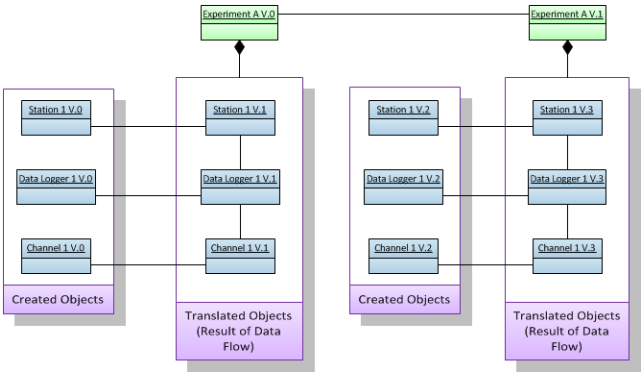

Figure 6: Logical representation of a subset of provenance data. Two versions of the finer granularity objects exist as a result of data flow dependencies and the Cycle Avoidance algorithm. These extra nodes must be removed for clean visualization of the provenance.

formation is correct in its representation, but many of the objects not important to users and can obfuscate the data's lineage in the visualization. To provide a clearer representation of the provenance, we must remove the double-versions created via the translated objects as a result of Cycle Avoidance.

It is important to note how specific these parsed cases are. In the figure, the *Experiment* objects are missing the extra translation versions. This is because these *Experiments* are only linked 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 manually link objects via data flow dependencies. A situation like this increases the difficulty in parsing individual cases for visualization.

6.1 Types of visualization

ProvDMS's visualization is web-enabled using various available JavaScript visualization libraries such as the JavaScript InfoVis Toolkit (JIT) and Data Driven Documents (D3JS). We designed a few types of visualizations for ProvDMS:

- *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 as some point in their life cycle. Multi-versioned objects do not change their identification from one version to another. As only their version changes, the nodes are no longer 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 ones do — but uses a system of forces acting on each node to determine their positions. This makes the system feel more interactive as users have the ability to apply forces to nodes in the graphs by dragging them. Figures 7 and 8 demonstrate some interesting results of this type of visualization.

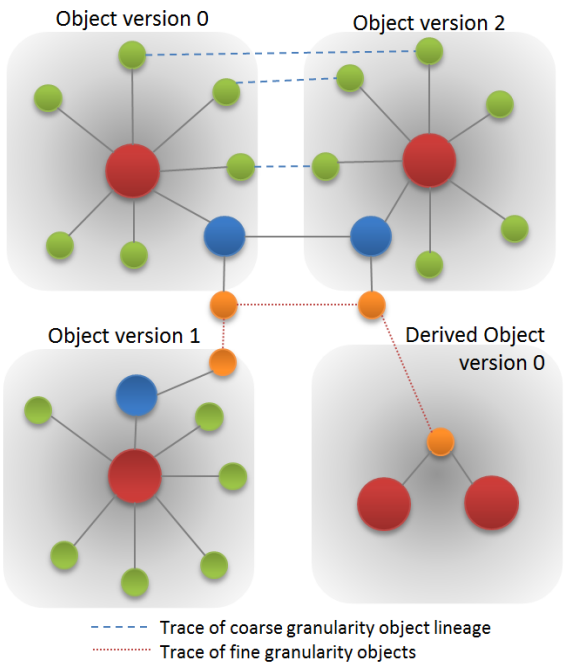

Figure 7: An illustrative view of the force-based node-link layout. A traceable flow of data lineage is visible, as well as a natural grouping of objects with similar granularity. Solid gray lines represent hierarchical connections between provenance objects that group together as information relevant to a single version of a user-defined or imported Experiment. Orange colored 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).

- *Unique, Contextual Node-Link Tree*: The current implementation of visualization in ProvDMS uses this approach in its provenance visualization module. Similar to the first, this approach 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 do. 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 9 is an example of this visualization.

7. CONCLUSION

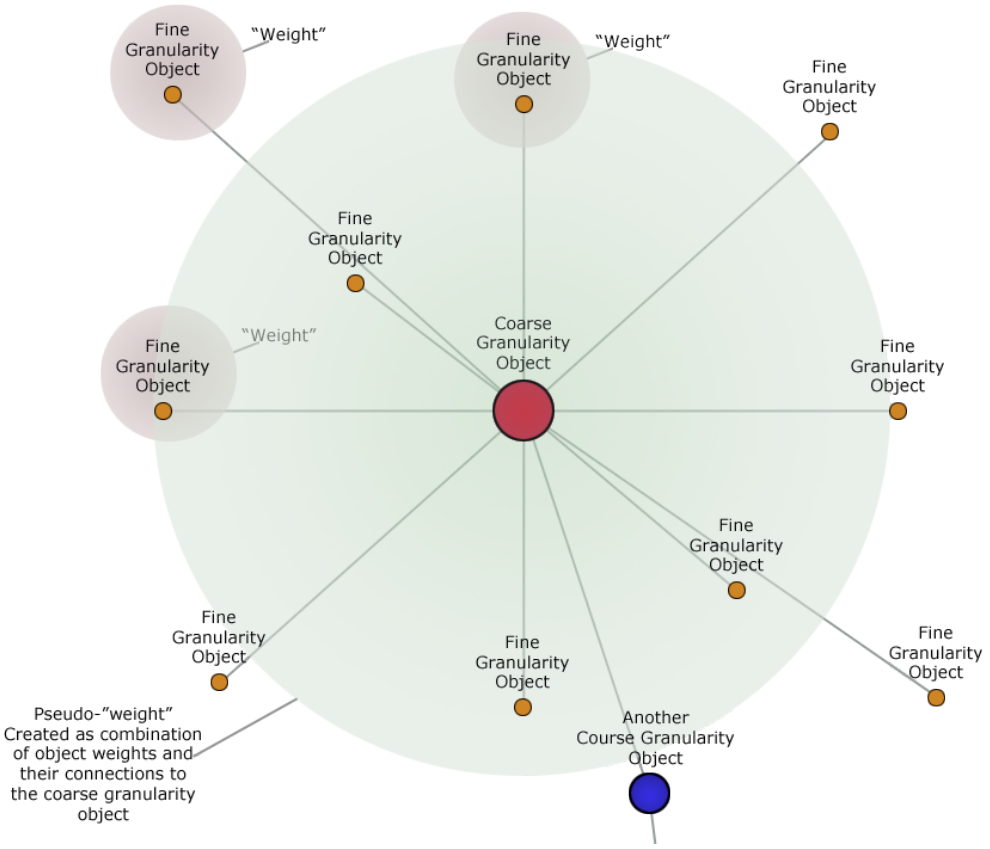

Figure 8: A close-up of one of the groupings in the force-based node-link layout. The innermost node and all of the finer-granularity nodes' connections create a pseudo "weight" that encompasses the entirety of the grouping of objects. Each object has its associated "weight" which affects the layout of all connected nodes. The grouping tends to act as a single node in the visualization.

Our attempts at bringing provenance to scientific research have highlighted some of the challenges and potential solutions for applying provenance to generalized data streams. Although we have been able to successfully build a system to handle provenance for ORNL's Flexible Research Platforms, this specific use case makes it less general than many other provenance systems. The availability of CPL as a library has been beneficial. Our successes with using CPL can be attributed to ProvDMS being independent of the provenance back-end providing us the required flexibility in system design. The C++ and PHP-wrapper code developed during the project was contributed back to the authors of the Core Provenance Library for future integration.

Research efforts are currently underway in automated sensor data validation, estimation for missing or corrupt data, and machine learning estimations of sensor health with plans to integrate workflows 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 guided and restricted the abilities of ProvDMS. We limited the features and the granularity of collected provenance to ensure minimal restrictions and little additional training required of the researchers. We believe we have succeeded in providing a simple interface for our users to manually keep track of their data and experiments. The modular design of ProvDMS allows us to add newer provenance collection methods as the system evolves. We expect improvements to soon follow using the knowledge from our experience with ProvDMS's design and use.

In the end, we hope ProvDMS can be an example of implementing and using provenance of a common data archetype in an environment normally devoid of information tracking methods. We also anticipate and hope that ProvDMS will demonstrate the power of such systems for enabling reproducible science.

8. ACKNOWLEDGEMENTS

The authors would like to thank Peter Macko and Margo Seltzer of Harvard University who are the authors of CPL for their continued support and guidance for 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

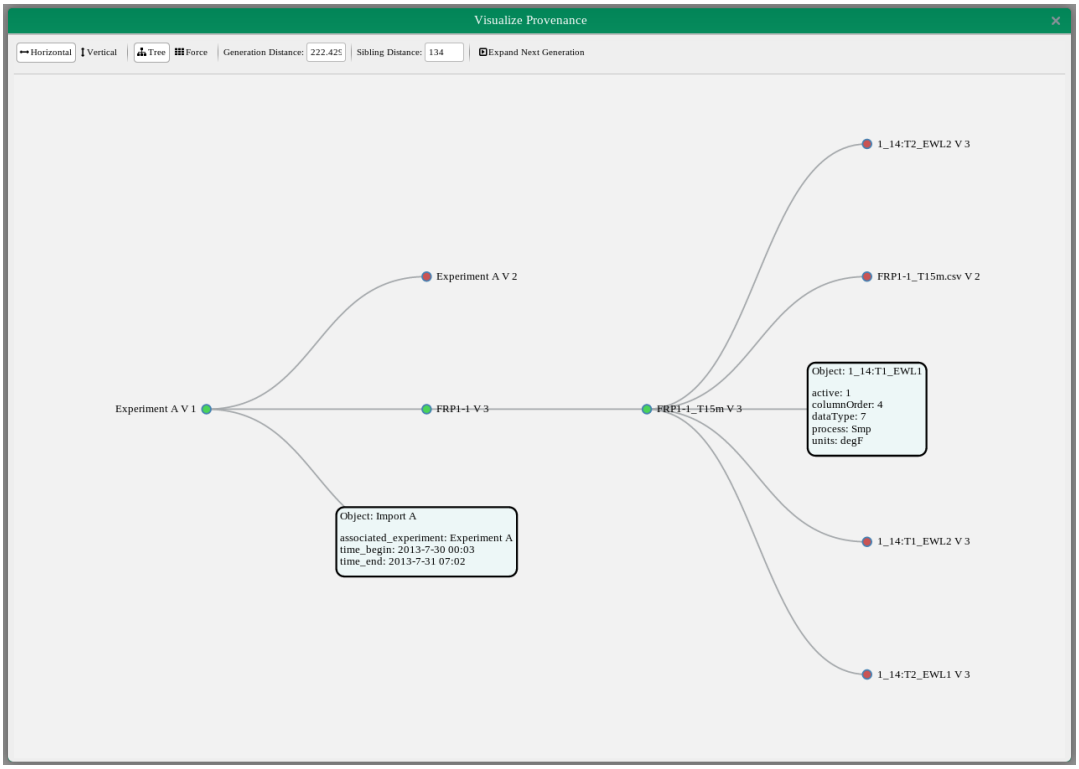

Figure 9: ProvDMS's current visualization, using Contextual Node-Link Trees. Two nodes are expanded to show meta-information at a finer granularity level than their parent nodes. Experiment nodes are the coarsest objects, while information specific to provenance objects, shown in rectangles, is at the finest granularity level.

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 United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

9. REFERENCES

[1] Sergio Alvarez, Javier Vazquez-Salceda, Tamás Kifor, László Z Varga, and Steven Willmott. Applying provenance in distributed organ transplant management. In *Provenance and Annotation of Data*, pages 28–36. Springer, 2006.

[2] Adriane Chapman and HV Jagadish. Issues in building practical provenance systems. *IEEE Data Eng. Bull.*, 30(4):38–43, 2007.

[3] Peter Macko and Margo Seltzer. A general-purpose provenance library. In *4th USENIX Workshop on the Theory and Practice of Provenance*, 2012.

[4] Carlos Scheidegger, David Koop, Emanuele Santos, Huy Vo, Steven Callahan, Juliana Freire, and Cláudio Silva. Tackling the provenance challenge one layer at a time. *Concurrency and Computation: Practice and Experience*, 20(5):473–483, 2008.

[5] Claudio T Silva, Juliana Freire, and Steven P Callahan. Provenance for visualizations: Reproducibility and beyond. *Computing in Science & Engineering*, 9(5):82–89, 2007.

[6] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *ACM Sigmod Record*, 34(3):31–36, 2005.

[7] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. Query capabilities of the karma provenance framework. *Concurrency and Computation: Practice and Experience*, 20(5):441–451, 2008.

[8] Martin Szomszor and Luc Moreau. Recording and reasoning over data provenance in web and grid services. In *On the move to meaningful Internet systems 2003: CoopIS, DOA, and ODBASE*, pages 603–620. Springer, 2003.

[9] Edward Tufte. Sparklines: theory and practice. *Retrieved September*, 13:2008, 2004.