The PBase Scientific Workflow Provenance Repository

Víctor Cuevas-Vicenttín
Universidad Panamericana - Guadalajara

Parisa Kianmajd
University of California at Davis

Bertram Ludäscher
University of California at Davis

Paolo Missier
Newcastle University

Fernando Chirigati
Polytechnic Institute of NYU

Yaxing Wei
Oak Ridge Natonal Laboratory

David Koop
Polytechnic Institute of NYU

Saumen Dey
University of California at Davis

Abstract

Scientific workflows and their supporting systems are becoming increasingly popular for compute-intensive and data-intensive scientific experiments. The advantages scientific workflows offer include rapid and easy workflow design, software and data reuse, scalable execution, sharing and collaboration, and other advantages that altogether facilitate “reproducible science”. In this context, provenance – information about the origin, context, derivation, ownership, or history of some artifact – plays a key role, since scientists are interested in examining and auditing the results of scientific experiments.

However, in order to perform such analyses on scientific results as part of extended research collaborations, an adequate environment and tools are required. Concretely, the need arises for a repository that will facilitate the sharing of scientific workflows and their associated execution traces in an interoperable manner, also enabling querying and visualization. Furthermore, such functionality should be supported while taking performance and scalability into account.

With this purpose in mind, we introduce PBase: a scientific workflow provenance repository implementing the ProvONE proposed standard, which extends the emerging W3C PROV standard for provenance data with workflow specific concepts. PBase is built on the Neo4j graph database, thus offering capabilities such as declarative and efficient querying. Our experiences demonstrate the power gained by supporting various types of queries for provenance data. In addition, PBase is equipped with a user friendly interface tailored for the visualization of scientific workflow provenance data, making the specification of queries and the interpretation of their results easier and more effective.

Received 28 October 2013 | *Accepted* 26 February 2014

Correspondence should be addressed to Víctor Cuevas-Vicenttín, Calzada Circunvalación Poniente No. 49 Ciudad Granja 45010 Zapopan, Jalisco, Mexico. Email: victorcuevasv@gmail.com

An earlier version of this paper was presented at the 9th International Digital Curation Conference.

Introduction

The origin and processing history of an artifact is known as its provenance. Data provenance is an important form of metadata that explains how a particular data product was generated, including the system and the steps in the computational process involved, the user responsible for its execution, time, and resources used, such as parameter settings, input data and software tools. Provenance information provides transparency and helps to audit processes and interpret data products. Common uses and applications of provenance include quality control, data curation, debugging, data discovery, and generally, the validation, attribution and reproducibility of scientific results.

The state of the art of scientific workflow systems such as Kepler (Ludäscher et al., 2006), Taverna (Wolstencroft et al., 2013) and VisTrails (Freire et al., 2006) provide controlled environments for specifying and enacting complex computational pipelines for which provenance information is automatically captured by the system in the form of traces (albeit often in proprietary formats). However, they lack the capabilities to enable the users to effectively query and visualize the provenance traces associated with a particular workflow. In addition, they do not address the need of scientists to share their workflow-based computational experiments for the scrutiny and benefit of the rest of the community. Therefore, the need arises for a repository allowing multiple users to store and query scientific workflow provenance in an interoperable manner. In this work we document the task of developing and putting into use such a repository, called PBase. Our ultimate goal is to incorporate such a repository into DataONE[1], a large scale and federated data infrastructure serving the Earth sciences community.

In order to build PBase we face three main challenges:

1. Defining a standard model for workflow provenance representation that is compatible with the main scientific workflow systems.
2. Characterizing the required functionality in terms of specific queries.
3. Developing the infrastructure that enables users to evaluate such queries efficiently, while also being easy to use for the scientific community.

We briefly describe next how we addressed those challenges. First, describing ProvONE, our proposed standard for scientific workflow provenance data, after which we present representative provenance queries. Subsequently, we describe the PBase architecture and how users interact with the system. Finally, we discuss related works, present our conclusions and discuss future work.

The ProvONE Model for Workflow Provenance

Scientific workflow systems provide a user-friendly graphical interface to specify a computational process in the form of a directed graph of interconnected tasks. Such a graph is an abstraction that can be regarded as prospective provenance, since it details the steps to follow in order to generate the desired result. In addition, as mentioned

1 DataONE: http://www.dataone.org/

earlier, the workflow system automatically captures the various events associated with the workflow execution, which is usually referred to as retrospective provenance. The full potential of provenance information is achieved when combining both. To this end, in the context of the DataONE project, an extension of the W3C PROV standard called ProvONE has been developed.

PROV (W3C PROV Working Group, 2013) provides a basic and standardized model for the representation and exchange of provenance information. ProvONE uses and extends the information elements of PROV to describe both workflow-level and trace-level information. The adoption of ProvONE establishes a common model for PBase and brings the advantages of interoperability with the emerging PROV standard. Support for ProvONE, serialized as XML, was recently added to VisTrails and other workflow systems can be supported through corresponding wrappers as well.

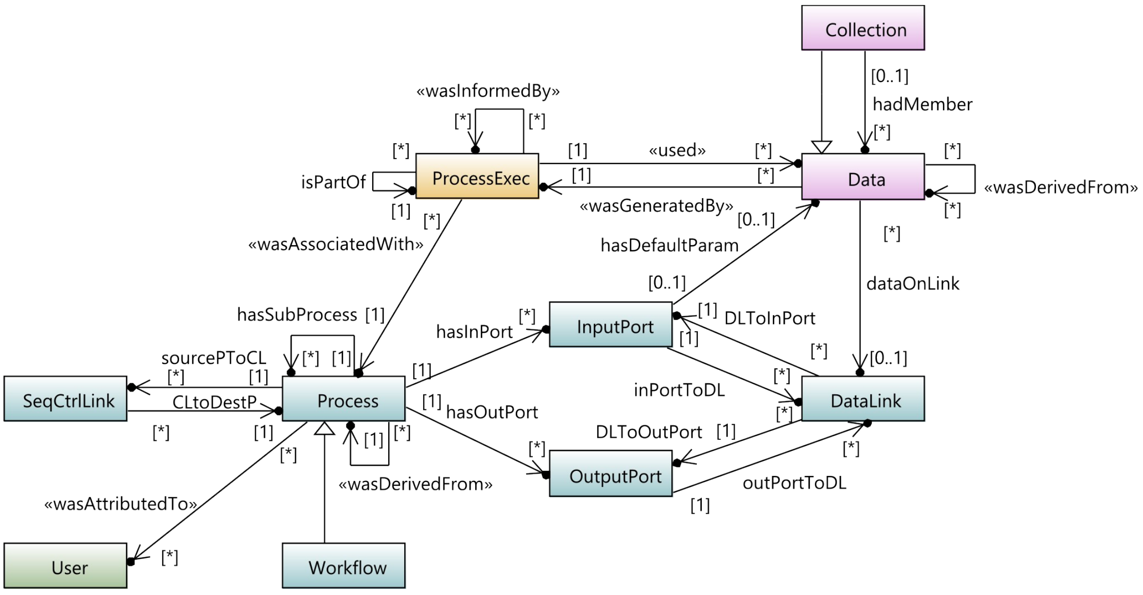


Figure 1. ProvONE conceptual model UML class diagram.

ProvONE Conceptual Model

The ProvONE conceptual model is illustrated by the UML diagram of Figure 1. All classes have a correspondent PROV type denoted by a UML stereotype (e.g. «entity»), whereas this is the case for only a subset of the associations (e.g. «used»).

The various tasks that form part of a workflow are represented by the *Process* class. *Processes* can be either atomic or composite, the latter case specified through the *hasSubProcess* self association. A given *Process* can be distinguished as a *Workflow*. Each *Process* has a series of *InputPorts* and *OutputPorts*, while the ports from the various *Processes* are connected through *DataLinks*. Note that both input and output ports can be associated with multiple *DataLinks*, thus allowing workflow models in which a single output is copied and sent to multiple destinations, as well as in which tasks take inputs from different sources through a single input port.

In order to specify executable instances of a *Workflow*, default parameters can be defined for some of its constituting *Processes*. The default parameters are represented by *Data* elements, which can be of various types (e.g. XML, JSON, files, etc.) including collections thereof. In addition, sequential control links can be specified between

Processes, as denoted by the *SeqCtrlLink* class, which enforce that a given *Process* can only be executed after the other *Process* that shares the link has sent the required signal. Finally, a particular *Process* that specifies a *Workflow* or part of it can be associated with a *User* that assumes responsibility for its creation. A detailed presentation of the ProvONE model based on an OWL-2 ontology can be found in DataONE Provenance Working Group (2014). An example of a concrete workflow and its corresponding trace is presented in the next section.

ScientificWorkflow Provenance Queries

ProvONE data is intrinsically graph-oriented, since workflows are represented as graphs having processes as nodes and traces in part as graphs whose nodes correspond to data entities (inputs and outputs) and computational activities (process executions). Therefore, the queries of interest over scientific workflow provenance correspond in many cases to queries over graphs, which have been extensively studied and employed in various domains.

Figure 2a presents an example workflow whose goal is to summarize ecological spatiotemporal data. The left branch calculates the standard deviation of the regional net ecosystem exchange data obtained from its input, and generates a map with one degree resolution illustrating the standard deviation of the data across the region. The right branch uses the same data to calculate monthly averages and presents the results in a plot. A trace generated by running this workflow is depicted in Figure 2b, with data items as ellipses and process executions as rectangles (some data and process execution nodes have been omitted for clarity).

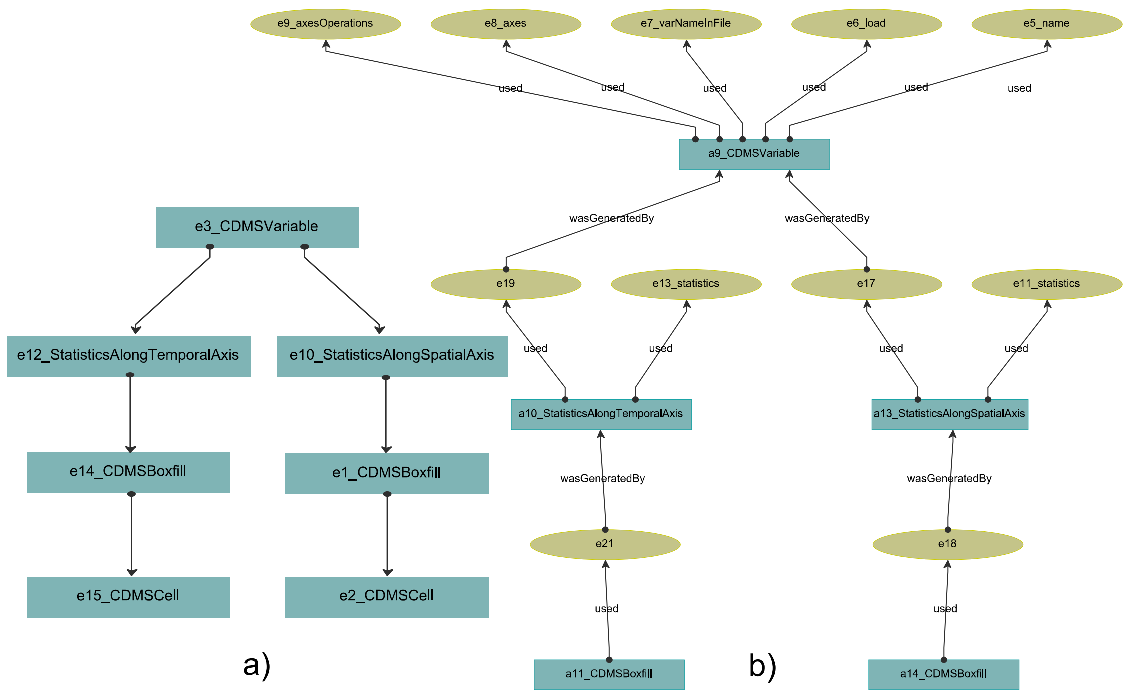


Figure 2. Example workflow (a) and corresponding trace (b).

In cooperation with climate scientists that employ scientific workflows, such as the one presented in Figure 2a, in their research we identified a suite of queries that are relevant for scientific workflow provenance. The queries in the suite can be classified in four main types:

1. *Lineage* queries, i.e., dealing with the derivation of data products. For example, "what are the datasets involved in the generation of this plot?".
2. *Execution analysis* queries, for example, "find the processes in a workflow that were not completed".
3. *Search* queries, for example, "find all the workflows that used a bilinear regrid module".
4. *Statistical* queries, such as "list the most used modules across all workflows".

Table 1 presents six queries that are incorporated into our implementation and their query type. Queries 1-3 can also be characterized as focusing on workflows, while queries 4-6 focus on traces. This distinction becomes relevant for user interaction as discussed in the next section.

Table 1. Example queries for scientific workflow provenance data.

Number	Query	Type
1	Compute the number of invocations of process `e7_Regrid`	Statistical
2	Find all inputs of a workflow across executions	Lineage
3	Find the modules of processes that were not executed	Execution
4	Find the process executions that were not completed	Execution
5	Find the processes that used data item `e10_regrid_method`	Search
6	Find the data products influenced by module `e7_Regrid`	Search

The PBase Provenance Repository

Provenance Data Storage and Querying

The challenges involved in developing the infrastructure capable of evaluating the aforementioned queries are twofold. First, we need an expressive query language covering diverse types of queries. Second, query evaluation should be efficient considering that provenance traces can be significantly large. We opted to base our infrastructure on the NoSQL graph database Neo4j,[2] following the rationale that it is customized for graph data and that it has recently incorporated Cypher, a declarative graph query language.

2 Neo4j: http://www.neo4j.org/

Table 2. Cypher specification of the queries in Table 1.

Number	Query
1	`START n = node:node_auto_index(name="e7_Regrid")`
	`MATCH m-[:wasAssociatedWith]-n`
	`RETURN count(m)`
2	`START n=node(*)`
	`MATCH (n)<-[:used]-()`
	`WHERE not ((n)-[:wasGeneratedBy]->()) AND HAS(n.wfID)`
	`  AND (n.wfID="wf1")`
	`RETURN DISTINCT n`
3	`START n=node(*)`
	`MATCH n<-[:wasAssociatedWith]-m`
	`WHERE HAS(n.vtType) AND HAS(n.wfID) AND HAS(n.runID)`
	`  AND n.vtType="vt:module_exec" AND n.completed=-1`
	`  AND n.wfID="wf1" AND n.runID="ex1"`
	`RETURN m`
4	`START n=node(*)`
	`WHERE HAS(n.completed) AND n.completed=-1 AND HAS(n.wfID)`
	`  AND n.wfID="wf1"`
	`RETURN DISTINCT n`
5	`START n = node:node_auto_index(name="e10_regrid_method")`
	`MATCH n<-[:used]-a`
	`RETURN distinct a`
6	`START n=node(*)`
	`MATCH n-[*]->a`
	`WHERE n-[:wasGeneratedBy]->() AND HAS(n.wfID) AND n.wfID="wf1"`
	`  AND HAS(a.module) AND a.module="e7_Regrid"`
	`RETURN DISTINCT n`

Table 2 presents the Cypher specification of the queries presented in the previous section. The basic structure of a Cypher query comprises a `START` clause that defines a series of starting points for graph exploration, a `MATCH` clause that specifies a pattern to be matched in the graph, a `WHERE` clause specifying additional filter conditions, and a `RETURN` clause that outputs the results, which can consist of complete nodes and edges or only of attributes thereof. A detailed discussion of the language is presented in Neo Technology Inc. (2014).

In some cases, however, the performance of Neo4j was found to be unsatisfactory when querying large traces. An alternative to address this issue is to incorporate new indexing and encoding techniques into Neo4j to achieve better performance. For reachability queries, a particular case of lineage that determines whether there is a directed path between a pair of nodes, we achieved major improvements by incorporating the tree cover encoding introduced in Agrawal, Borgida and Jagadish (1989) to the stored workflows and traces. This form of encoding can be generated in quadratic time on the number of nodes and in the worst case can be of a linear size for some nodes. However, in practice its size is significantly lower and enables the system to answer reachability queries in nearly a constant time. For a test graph of 10,000 nodes the encoding enabled us to evaluate 8,000 random queries in under a minute on a laptop, while it took more than two hours without it.

System Architecture

The architecture of our current prototype is presented in Figure 3. The system adopts a Java platform three tier architecture with a client application running on a Web browser, while the server runs on the Tomcat application server. The server in turn is separated into the application logic and the database, the latter supported by Neo4j 1.9.1 running on embedded mode. Currently, our prototype only enables the creation of a database from a VisTrails exported XML trace file, which contains the workflow specification and the various traces corresponding to the workflow runs. This file is submitted to PBase through the *TraceUpload* component of the Web client as an HTTP POST request, which includes a user supplied identifier for the database to be created.

The PBase server application implements a series of Web services to handle user requests. In the case of submitted traces, these are processed by the *ProvTraceUploadService* that stores the file temporarily on the server, after which it is processed by the *PROVBuilder* component in the database tier, which relies on the *GraphDAO* component to store the data in an actual Neo4j database. The graph data for Neo4j is complemented by the tree cover encoding discussed earlier that is generated by its corresponding component. One database is created for each original XML trace file with its supplied identifier.

The stored workflows and traces can be accessed and queried through the *PBaseGUI*, whose generated requests are handled by a series of restful Web services (characterized as resources) in the PBase application. Requests are sent through HTTP GET, and are served through Cypher queries issued to the Neo4j database by the *GraphDAO* component, which also serializes the results in JSON to be sent back to the client.

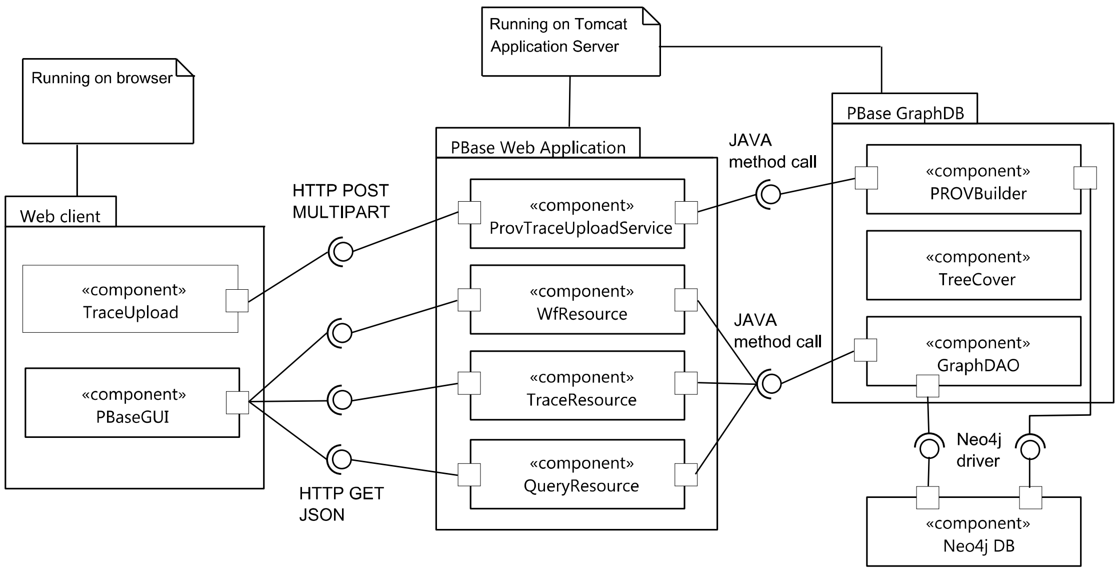


Figure 3. PBase System Architecture.

User Interaction with PBase

As a proof of concept user interface aimed at end users, we developed a Web-based GUI that enables users to upload a provenance trace, visualize the workflow alongside its various traces, issue queries and obtain visualizations of their results. Our experience with climate scientists shows that this makes PBase much easier to use in comparison to the generic tools included in Neo4j, particularly for users that are not familiar with scientific workflow provenance in the first place. The Web-based GUI was developed with the YUI 3.14.1 framework[3] and the mxGraph 2.3.0.5 library[4], a snapshot of it is depicted in Figure 4. Next we describe the main aspects of its functionality.

Workflow and trace loading visualization.

In order to load a workflow and its corresponding traces from the database, the user supplies the identifier specified when the trace was uploaded. Once the workflow and traces have been loaded, a visualization of the workflow is generated on the left hand side and of the trace on the right hand side. If multiple traces are associated with the workflow, these can be changed from a menu on the load tab on the right-hand side panel.

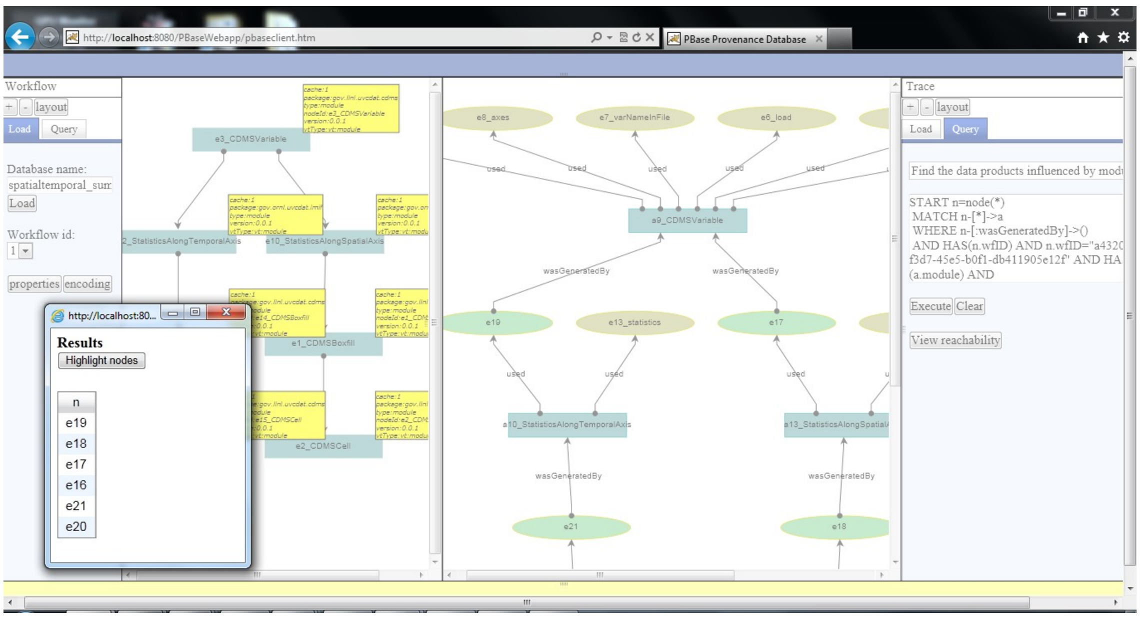
The size of the visualization panels of the workflow and the trace can be changed in order to focus on one or the other if desired. Their respective graph visualizations can also be zoomed in or out and dragged independently. Initially, only the identifiers of the various process, process execution and data nodes are displayed, but the additional properties associated with the nodes can be visualized by overlaying text boxes that can be shown or hidden as required. The tree cover encoding can also be visualized in an analogous manner.

Workflow and trace querying and results visualization.

The user can select from a suite of example queries based on their textual description, such as Queries 1-3 from Table 1. Once the query is selected from the menu, its corresponding Cypher specification appears on the accompanying text area in the querying tab panel, in our example these would be queries 1-3 from Table 2. The Cypher specifications are generated automatically to include the identifiers of the loaded workflow and trace. The user can then execute the query directly or edit is as required. Alternatively, she can also specify an entirely new query from the basic query structure. This functionality is present for both the workflow and the trace in its corresponding panels on the GUI.

After the query is executed its results are presented in tabular form on a pop-up window, as shown on the left hand side of Figure 4. If node attributes form part of the result these are presented directly in the table, whereas if entire nodes are part of the result their identifier is shown while the actual nodes can be highlighted on the corresponding graph if desired. Since lineage is particularly important for provenance data, PBase offers the capability to visualize the reachability of all the nodes in a workflow or a trace with respect to a particular node. Concretely, the user can select a node on the workflow or trace graph and the descendants and ancestors of that node will be highlighted using different colors (no change occurs if a node is neither an ancestor nor a descendant). Such highlighting is performed entirely on the client side via the GUI by taking advantage of the encoding pre-computed on the server, thus resulting in a more fluid user interaction.

3 YUI 3.14.1: http://yuilibrary.com/
4 MxGraph 2.3.0.5: http://jgraph.github.io/mxgraph/



n
e19
e18
e17
e16
e21
e20

Find the data products influenced by modu

```
START n=node(*)
 MATCH n-[*]->a
 WHERE n-[:wasGeneratedBy]->()
 AND HAS(n.wfID) AND n.wfID="a4320
f3d7-45e5-b0f1-db411905e12f" AND HA
(a.module) AND
```

Figure 4. PBase Web-based graphical user interface.

Related Work

Regarding the ProvONE model, independent extensions of PROV for scientific workflows are presented in Costa et al. (2013) as well as in Belhajjame et al. (2013) (focusing on workflow preservation). Although ProvONE may be in its core very similar to these models, we additionally developed a repository with querying and visualization capabilities representing a more complete and flexible infrastructure. Our work in part complements previous results presented in Missier et al. (2012), which focused on enabling the specification of mappings between data produced and used in different traces. This paper focuses instead on enabling declarative querying and effective visualization. In particular, we believe that the adoption of a graph query language can make the system easier to use as opposed to the approach presented in Marinho et al. (2012) using Prolog, for example.

Future Work

Future work involves enhancing the capabilities of PBase with queries that cannot be evaluated in Cypher or other graph query languages, such as finding workflows and traces by graph similarity or performing sophisticated keyword searches that take ranking into account. Integration with the DataONE Cyberinfrastructure is also envisaged to enable scientists to publish, search, explore and query scientific data provenance in DataONE. Support for other workflow systems is also planned to make the system more practical to the scientific community. Although our experience with Neo4j was satisfactory overall, in particular regarding the expressivity of the Cypher query language, we are also exploring RDF/SPARQL graph databases as an alternative. PROV is inclined towards these Semantic Web standards and consequently, RDF/SPARQL are being adopted in the most recent ProvONE specification.

Conclusions

Provenance data is of foremost importance in eScience, where scientific workflow management systems already play a key role. In order to take advantage of scientific workflow provenance data it is essential to be able to share this data; the ProvONE model makes this feasible by extending PROV with workflow-specific concepts. In addition, it is necessary to be able to store and query such data efficiently, while also being able to do so in a user friendly manner and accompanied by adequate visualizations. PBase fulfills these requirements by adopting a powerful graph database and offering a user interface tailored for scientific workflow provenance. Our experiences show that the functionality offered by PBase represents an important first step for the wider adoption of scientific workflow provenance as part of the computer-based scientific process.

Acknowledgements

The authors would like to thank the members of the DataONE EVA Working Group for their collaboration in the specification of the climate data analysis workflows, and of the Provenance Working Group in the specification of the ProvONE model and the functionality of PBase. The first author was affiliated with the University of California at Davis at the time this research was conducted. This work was supported by NSF Award OCI-0830944 (DataONE).

References

Agrawal, R., Borgida, A. & Jagadish, H.V. (1989). Efficient management of transitive relationships in large data and knowledge bases. *ACM SIGMOD Record, 18*(2), 253–262. doi:10.1145/66926.66950

Belhajjame, K., Klyne, G., Garijo, D., Corcho, O., García-Cuesta, E. & Palma, R. (2013). Wf4Ever research object model. Retrieved from https://w3id.org/ro/

Costa, F., Silva, V., de Oliveira, D., Ocaña, K., Ogasawara, E., Dias, J. & Mattoso, M. (2013). Capturing and querying workflow runtime provenance with PROV: A practical approach. In K.-U. Sattler, D. Bianchini, M. Castellanos, F. Daniel, R. De Virgilio, V. De Antonellis, … A. Gounaris (Eds.), *Proceedings of the Joint EDBT/ICDT 2013 Workshops* (pp. 282–289). New York, NY: ACM. doi:10.1145/2457317.2457365

DataONE Provenance Working Group. (2014). ProvONE: A PROV extension data model for scientific workflow provenance. Retrieved from http://purl.org/provone

Freire, J., Silva, C.T., Callahan, S.P., Santos, E., Scheidegger, C.E. & Vo, H.T. (2006). Managing rapidly-evolving scientific workflows. In I. Foster (Ed.), *Lecture Notes in Computer Science: Vol. 4145. Provenance and Annotation of Data* (pp. 10–18). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/11890850_2

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., … Zhao, Y. (2006). Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience 18*(10), 1039–1065. doi:10.1002/cpe.994

Marinho, A., Murta, L., Werner, C., Braganholo, V., Cruz, S.M.S.d., Ogasawara, E. & Mattoso, M. (2012). Provmanager: A provenance management system for scientific workflows. *Concurrency and Computation: Practice and Experience 24*(13), 1513–1530. doi:10.1002/cpe.1870

Missier, P., Ludäscher, B., Dey, S.C., Wang, M., McPhillips, T.M., Bowers, S., … Altintas, I. (2012). Goldentrail: Retrieving the data history that matters from a comprehensive provenance repository. *International Journal of Digital Curation 7*(1), 139–150. doi:10.2218/ijdc.v7i1.221

Neo Technology, Inc. (2014). Neo4j Cypher Refcard 1.9. Retrieved from http://docs.neo4j.org/refcard/1.9/

W3C PROV Working Group. (2013). *PROV-Overview: An overview of the PROV family of documents.* Retrieved from World Wide Web Consortium website: http://www.w3.org/TR/prov-overview/

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., … Goble, C. (2013). The Taverna workflow suite: Designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research, 41*(W1), W557–561. doi:10.1093/nar/gkt328