Linking Design-Time and Run-Time: A Graph-based Uniform Workflow Provenance Model

Xiaoyi Duan, Jia Zhang, Qihao Bao
Department of Electrical and Computer Engineering
Carnegie Mellon University – Silicon Valley
{xiaoyi.duan, jia.zhang, qihao.bao}@sv.cmu.edu

***Abstract*—Workflow is an important way to mashup reusable software services to create value-added data analytics services. Workflow provenance is core to understand how services and workflows behaved in the past, which knowledge can be used to provide better recommendation. Existing workflow provenance management systems handle various types of provenance separately. A typical data science exploration scenario, however, calls for an integrated view of provenance and seamless transition among different types of provenance. In this paper, a graph-based, uniform provenance model is proposed to link together design-time and run-time provenance, by combining retrospective provenance, prospective provenance, and evolution provenance. Such a unified provenance model will not only facilitate workflow mining and exploration, but also facilitate workflow interoperability. The model is formalized into colored Petri nets for verification and monitoring management. A SQL-like query language is developed, which supports basic queries, recursive queries, and cross-provenance queries. To verify the effectiveness of our model, A web-based, collaborative workflow prototyping system is developed as a proof-of-concept. Experiments have been conducted to evaluate the effectiveness of the proposed SQL-like graph query against SQL query.**

Keywords-scientific workflow, integrated provenance model, design-time provenance, run-time provenance

I. INTRODUCTION

As the rapid development of big data, data-oriented workflows (or so called mashups) have become an important way to realize and streamline different tasks in the life cycle of big data analytics, such as data integration, data mining, and data visualization. In order not to recreate the wheels, existing data processing modules can be wrapped up as reusable services (i.e., APIs) in building value-added workflows. Thus, it is important to understand how existing data analytics tools/modules have been used, so as to provide context-aware mashup/workflow recommendation [1]. Provenance management is core to examine past usages of services and explore possible mashup composition, in addition to helping people gain insights into reasoning, verify experiments and reproduce the results [2].

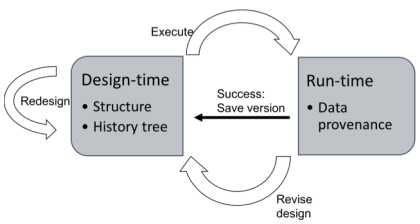


Fig. 1. Workflow provenance in two phases.

To date, workflow provenance management is generally divided into three categories: retrospective provenance, prospective provenance, and evolution provenance [3]. Retrospective provenance (*r-prov*) captures past workflow execution and data derivation, also called data provenance [4]. Prospective provenance (*p-prov*) captures representation of workflow structure [5, 6]. Evolution provenance (*e-prov*) captures how a workflow has become what it is now, also called provenance on process [5].

Putting the three forms of provenance side-by-side as shown in Fig. 1, we can see that they are actually captured at two phases during a workflow lifecycle: design time and run time. Prospective and evolution provenance are both captured at design time, which can be used to derive contributors, workflow structures and all historical designed versions. Retrospective provenance is captured at run time, which can be used to analyze data artifacts from different processes of workflows. Therefore, we categorize workflow provenance into design-time provenance and run-time provenance. Design-time provenance can be further categorized into structure-oriented provenance (i.e., prospective provenance) and action-oriented provenance (i.e., evolution provenance).

Due to the exploratory nature of data-intensive study, researchers constantly move back and forth between the two worlds. Design-time and run-time provenance is in turn updated by these activities, as shown in Fig. 1. When researchers edit a workflow during design-time, all related operations are recorded in a history tree (*e-prov*). After the workflow is designed, it will be executed and data provenance (*r-prov*) will be generated and recorded. If the execution fails, the researchers may go back to revise the workflow and start another round of iteration. If the execution succeeds, the structure of the workflow (*p-prov*) will be persisted as a version. Even if an execution is successful, the researchers may also want to explore various paths, thus go back to design-time.

During such iterative transitions between the two worlds, researchers may at any time wish to check among different types of provenance. For example, observing an interesting data product (*r-prov*) may trigger workflow developers to review corresponding workflow design (*p-prov*) and what design changes (*e-prov*) from an earlier version lead to the outcome. They may also rerun the workflow (*p-prov*) to verify the reproducibility of the workflow (*r-prov*), and tune the workflow (*e-prov* and *p-prov*) to explore various possibilities.

Such a typical data science scenario calls for an integrated view of provenance and seamless exploration transition

among different types of provenance, that is, cross-provenance query. However, no existing model or system collects all three forms of provenance. PROV [7] framework, endorsed by the World Wide Web Consortium (W3C), focuses on run-time provenance. Most Scientific Workflow Management Systems (SWFMSs) only support one or two types of provenance. For example, Kepler [8] and Taverna [9] store run-time provenance; Trident [10] and Data-One [11] store run-time and prospective provenance. VisTrails [4] stores run-time provenance and evolution provenance, while prospective provenance is implied by rebuilding workflow through history tree. In addition, existing SWFMSs store different types of provenance separately supported by different query methods. As a result, cross-provenance queries cannot be easily supported.

To address these issues, we make three contributions in this paper. First, we propose a uniform workflow provenance model, aiming to bridge between workflow design-time and run-time. Without recreating the wheel, we have significantly extended the standard PROV model to combine various types of provenance in one unified model and store them in one graph. Second, we have formalized our provenance model using Colored Petri nets. Third, we have designed a SQL-like language to query workflow provenance based on our model, which supports basic queries, recursive queries, and cross-provenance queries. In addition, to verify the effectiveness of our model, we have implemented a web-based, multi-user collaborative workflow prototyping system. To the best of our knowledge, we are the first effort to integrate all three forms of provenance into one uniform model, supported by formalization and graph-based query language and mechanism. Such a uniform provenance model will also facilitate workflow interoperability.

The remainder of the paper is organized as follows. Section II discusses related work on data models for provenance. Section III introduces our uniform data model integrating three types of provenance generated in either design-time or run-time. Section IV formalizes our data model using Petri nets. Section V presents provenance store, query and recommendation based on our model. Section VI describes a prototyping system and reports our experimental results. We draw conclusion in Section VII.

II. RELATED WORK

Provenance helps to ensure reproducibility and sharing of scientific workflows. Therefore, provenance management has been widely acknowledged as a critical functionality for any Scientific Workflow Management Systems (SWFMSs) to capture and manage workflow information [11, 12]. There are three forms of provenance: prospective provenance (p-prov), retrospective provenance (r-prov) and provenance of process (e-prov). They respectively capture workflow specification, detailed log of workflow execution and workflow evolution. Various SWFMSs have different focuses on the provenance types [4-6]. Kepler [8] implements a provenance recorder to track information about workflow runs. Taverna [9] uses event logs to record data provenance, and adopts Semantic Web technologies to represent provenance metadata. VisTrails [4] records provenance for workflow evolution and data products [12]. In addition, several stand-alone provenance systems have been developed, including the PReServ system developed under the Provenance Aware Service Oriented Architecture (PASOA) project [13] and the Karma system [14].

Because retrospective provenance does not directly depend on prospective provenance, most systems separately store retrospective provenance and prospective provenance. For example, Taverna uses SCUFL language to describe p-prov, while using RDF to store r-prov. VisTrails uses XML to capture p-prov, while use relational database to store r-prov [4]. Few systems support provenance of process [9] except VisTrails [4]. In contrast, we propose to build a uniform provenance model and use the uniform language to process provenance.

To promote the interoperability of provenance among different workflow systems, the Open Provenance Model (OPM) was initiated in 2007 [15]. In recent years, PROV [7] framework, endorsed by the World Wide Web Consortium (W3C), formalizes inter-operable interchange of provenance information in heterogeneous environments. A number of emerging applications use either OPM or PROV model for capturing provenance traces [16, 17]. While OPM and PROV only represent retrospective provenance, D-PROV [11], ProvOne [18], and P-PLAN [19] have all proposed extensions to the PROV model for workflow structure. In contrast, we have extended PROV into an integrated data model carrying both design-time and run-time provenance.

Some researchers focus on provenance storage and develop techniques to optimize query over the provenance models. Heinis and Alonso create an interval-based representation for provenance storage [29]. Chapman et al. propose a set of factorization processes and inheritance-based methods to reduce the size of actual provenance datasets [30]. To facilitate focused query and navigation over large amounts of provenance, Biton et al. develop a provenance abstraction technique to return only relevant and abstracted provenance information to a user [20].

To support provenance query, provenance information can be extracted from log files generated by SWFMSs [4-6], and then stored into a relational database for query [21-23]. Gadelha Jr et al. [24] propose query patterns to simplify query design. Anand et al. [25] propose a query language to query both lineage and structures on provenance graph and then store in a relational database. Because queries on evolution provenance focus on relationships, we decided to use graph database (neo4j, https://neo4j.com/) to store all forms of provenance, and query using the Cypher language (https://neo4j.com/developer/cypher-query-language).

III. POVENANCE MODEL

As shown in Fig. 1, researchers constantly visit back and forth between design-time and run-time during the development of a workflow. In order to provide seamless cross-provenance query, we propose a uniform provenance model carrying prospective provenance, evolution provenance, and retrospective provenance into one integrated model. Instead of recreating the wheel, we have extended the

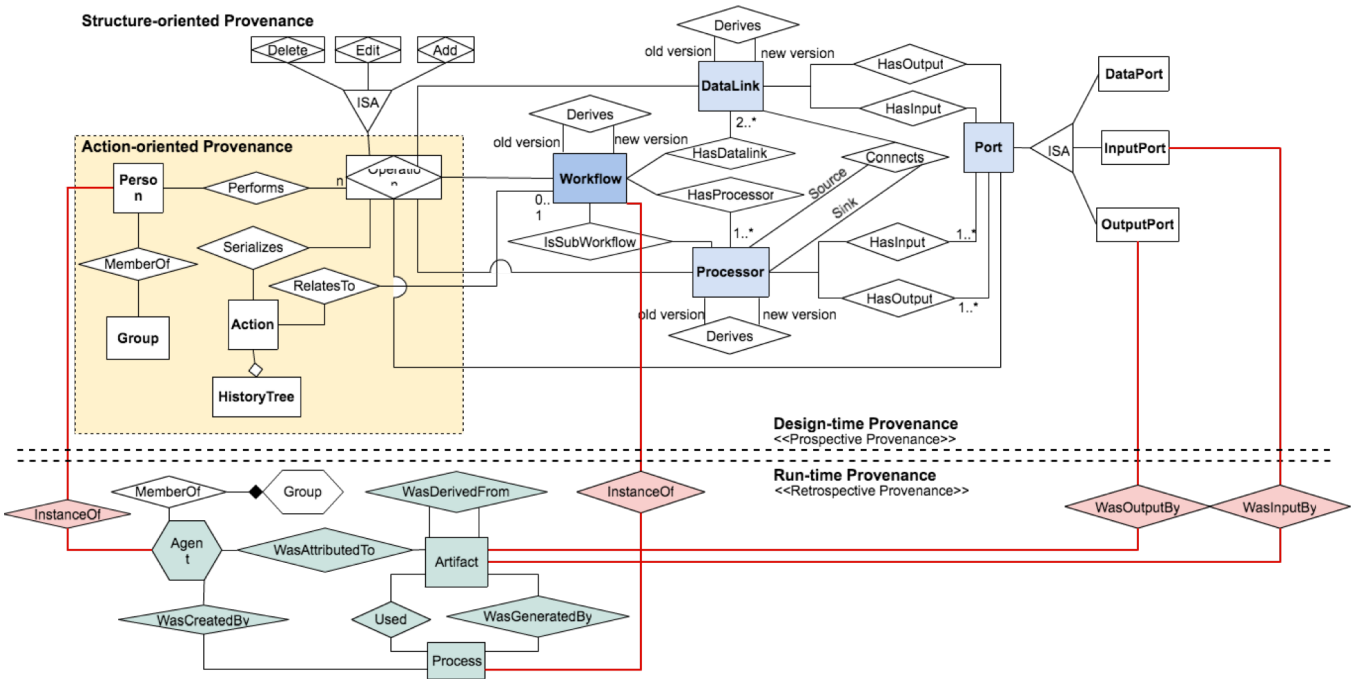


Fig. 2. Provenance Data Model

ad hoc standard PROV [21] model to build our unified provenance model. Fig. 2 illustrates our provenance model using as an ER model.

The overall model is divided into design-time provenance (upper portion) and run-time provenance (lower portion) separately by lines. As shown in Fig. 2, the two worlds are connected through four portals.

A. Design-time Provenance

During workflow design time, provenance is generated in two formats, which are Action-Oriented Provenance and Structure-Oriented Provenance. Action-Oriented Provenance records history of workflow design actions, while Structure-Oriented Provenance captures evolution of workflow structure. Action-oriented provenance can be organized as a tree structure where each node is an action and each path is an action sequence. Note that the structure can be obtained by traversing a path in the action history, however, it is more efficient to retrieve workflow structure directly. Therefore, our provenance model stores both of them.

(1) Action-Oriented Provenance

Action-Oriented Provenance (AOP) records action history in designing workflow. The ER model of AOP is on the left-hand side of Fig. 2, delimited in a grey box. Action history forms a tree structure of provenance, where each node represents an action, and each edge connects two actions by their time partial relation. Hence, a path from the root node to any node in the history tree is related to a timestamp of the corresponding workflow. When a researcher implements an operation on a workflow, the operation is serialized to an action node in the AOP. Hence, AOP model includes five entity types (Person, Group, HistoryTree, Action and Operation) and four relation types (MemberOf, Performs, Serializes and RelatesTo). In addition, Operation entity has three types: add, delete and edit.

(2) Structure-Oriented Provenance

Structure-Oriented Provenance (SOP) captures the evolution of workflow structure. Although the PROV-DM only models workflow run-time provenance, some concepts can be used in design process, such as "derivation" and "attribution" relations. We apply them to represent the "derives" and "performs" relations in our model, respectively.

In general, SOP is composed by one or more processors and datalinks, some of which may in turn include sub-workflows. Processors are connected by data links through their input ports and output ports. Author information is caught separately, which is associated with every activity over any entity. Therefore, SOP model includes eight entity types, Person, Group, Workflow, DataLink, Processor, DataPort, InputPort and OutputPort, and six relation types, Derives, IsSubWorkflow, HasDataLink, HasProcessor, HasOutput, and HasInput. It should be noted that Operation is a relation in SOP but an entity in AOP. As for SOP, it represents three types of operations (Add, Edit and Delete) acting on the structure of a workflow.

B. Run-time Provenance

During workflow execution time, provenance captures past workflow execution and data derivation information. As mentioned above, the PROV-DM (PROV Data Model, https://www.w3.org/TR/prov-dm) is an *ad hoc* standard model for capturing run-time provenance. Therefore, we use PROV-DM to model run-time provenance, whose major components and relationships are presented in the bottom part of Fig. 2 and the same concepts with PROV-DM are

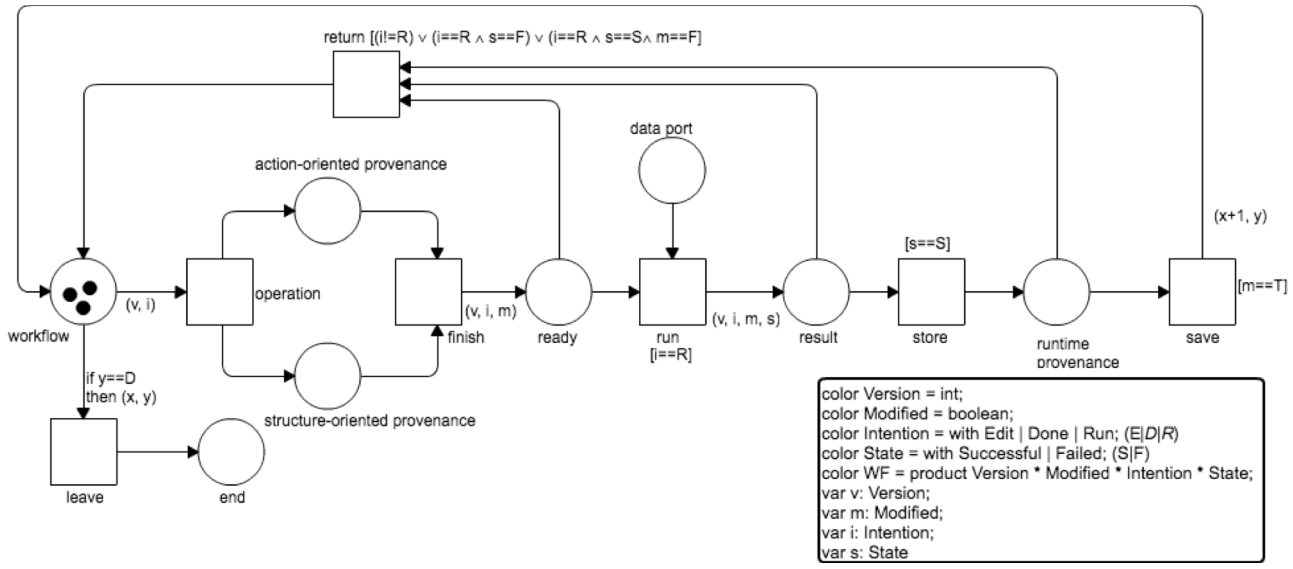


Fig. 3. Relation between Workflow Design and Execution

highlighted in grey. An artifact is an entity that can be either real or imaginary. A process refers to some software or procedure that acts upon or with artifacts. Therefore, the relationship between them is that processes utilize artifacts in producing artifacts. In other words, artifacts may be consumed, processed, or modified by processes. An agent is responsible for a process taking place.

In our model, there are four entity types (Process, Artifact, Agent and Group) and five relationship types (WasDerivedFrom, Used, WasGeneratedBy, WasCreatedBy and MemberOf).

C. Portals Linking Design-time and Run-time Worlds

To combine run-time provenance and design-time provenance, we locate intrinsic connections between them. As shown in Fig. 2, the two worlds are connected through four portals: (1) a person who designs a workflow acts as an agent at run-time; (2) a designed workflow is executed and creates some artifacts; (3) a workflow is designed to take as input an artifact results in the run-time; and (4) a workflow is designed to take as output an artifact results in the run-time.

In more detail, a workflow generated at design-time is executed as a process during the run-time. Therefore, "workflow" is the instance of "process." Similarly, a person who involves in design-time is the agent who executes the workflow, so "person" is the instance of "agent." Another important entity in run-time provenance is artifact, which is the output of an output port or the input of an input port.

IV. POVENANCE MODEL FORMALIZATION

In order to evaluate the correctness of our provenance model, we formalize it using petri nets. A Colored Petri Nets (CPN) is one type of graphical modelling language for concurrent systems. CPN allows tokens with a data value attached to them. This attached data value is called token color. Although the color can be of arbitrarily complex type, places in CPNs usually contain tokens of one type. This type is called color set of the place.

We define Workflow-level Colored Petri Net (WCPN): A *net* is a tuple $N = (P, T, A, \Sigma, C, N, E, G, I)$ where:

- P= {workflow, action-oriented provenance, structure-oriented provenance, ready, data port, result, runtime provenance, end} is a set of *places*.
- T= {operation, finish, run, store, save, end} is a set of *transitions*.
- Σ= {Version, Modified, Intention, State, WF} is a set of color sets defined within CPN model. WF is the combination (i.e., product in CPN) of other four colors.
- C is a color function. It maps places in P into colors in Σ.
- E is an arc expression function. It maps each arc into the expression e. The input and output types of the arc expressions must correspond to the type of the nodes the arc is connected to.
- G is a guard function. It maps each transition $t \in T$ to a guard expression g. The output of the guard expression should evaluate to Boolean value: true or false. There are four guards on four transitions, which are return, run, store and save.

WCPN is illustrated in Fig. 3 as a Petri net representation of the overall two worlds. First, workflow is a combined token ("WF") of four tokens, "Version," "Modified," "State," and "Intention." "Version" is the workflow version number; "Modified" is a bool token to show if the workflow is modified; "State" indicates whether execution is "Successful" or "Failed"; "Intention" indicates whether author wants to continue ("Edit" or "Run") or leave ("Done"). Either during the design process or after execution, "Intention" token can decide if the author will leave no matter what "State" is. Otherwise, workflow structure has to be persisted after each successful execution. Note that multiple arcs can connect the same pair of nodes with different arc expressions.

We model workflow-level petri-nets to cover scenarios in the real world. As shown in Fig. 2, we obtain traces and data provenance from the data model, then we can use petri-

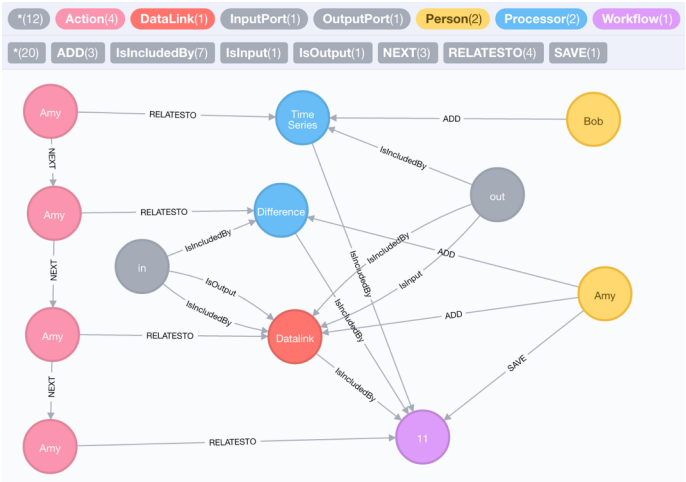


Fig. 4. Provenance storage in Neo4j graph database.

net (Fig. 3) to evaluate it.

V. PROVENANCE MANAGEMENT

A. Provenance Storage

Since provenance query focuses on relations, we store both run-time and design-time provenance into a graph, which can be noted as G={V, E} where V represents nodes of entity and E represents edges of relation. For example, in design-time provenance, V includes "Processor," "Datalink," "Workflow" and so on. The types of E are "IsIncludedBy" and "IsDerivedBy." In action-oriented provenance, V is type of action node and E represents temporal relation between two actions. Fig. 4 shows the representative graph of our model in a graph database.

B. Provenance Query

Based on the provenance data model, we pose the following provenance queries [27] that facilitate scientists to discover and analyze useful information in a workflow design and development process. We categorize queries into three types: (1) to analyze inner-workflow information, such as sub-structure, version-management; (2) to discover relationship in user-user network, such as frequent or potential collaboration; and (3) to find information in user-workflow, such as contribution evaluation. We give some sample queries for each type as below:

Q1: Inner-workflow queries:

a. *Show all the details about how W^{v3} has been designed and evolved as it is;*
b. *For workflow W^{v3}, which versions of comprising steps 1 and 2 are used? Who designed the two steps? How are they designed or refined? How are they merged?*
c. *What are the previous versions of W^{v3}? Why was it refined?*

Q2: user-user queries:

a. *Return all user pairs who designed some workflows collaboratively;*
b. *For user u, recommend a potential collaborator to work on a workflow.*

Q3: user-workflow queries:

a. *Return all the designers who contributed to the design of W^{v3};*
b. *Return the sub-workflows designed or refined by user s_1;*

Because a graph can provide information of various types of nodes and relations, it answers these queries by traversing paths in the graph. Q2b is not a basic query, so we will discuss it in the next part. Here we provide the following cypher code accordingly:

Q1a:

match (:Person)-[r]-()-[:IsIncludedBy]->(:Entity{id:"Wv3"}) return r order by r.time*

Q1b:

match (:Person)-[r]-(e)-[:IsIncludedBy]->(:Entity{id:"en3"}) return e.name, e.version*

Q1c:

match ()<-[r:IsDerivedBy]-(:Entity{id:"Wv3"}) return r*

Q2a:

match (p1)-[: ADD|EDIT|SAVE]->()-[:IsIncludedBy]->()<-[:IsIncludedBy*]-()<-[: ADD|EDIT|SAVE]-(p2) return p1,p2*

Q3a:

match (:Entity{id:"Wv3"})<-[r:IsIncludedBy]-()<-[:ADD|EDIT|SAVE]-(p) return p*

Q3b:

match ()-[r2:IsIncludedBy]-()<-[r1: ADD|EDIT]-(p:Person{name:"s1"}) return r1, r2

Additionally, to make graphical queries more friendly to users, we have designed a SQL-like language which is simple, flexible and effective to write queries. Cypher queries will be transformed into SQL-like queries. Therefore, scientists can use a familiar query language instead of learning a new one. Furthermore, they can avoid from writing redundant and complex queries that may contain multiple JOINs. For example, we convert some sample queries as below:

Q1a: Select structure
Where Name='workflow1', version='1.1'
Q2a: Select collaborator
Where Name='workflow1', version='1.1'
Q3a: Select contributor
Where Name='workflow1', version='1.1'

C. Recommendation

Storing both run-time and design-time provenance in graphs not only serves basic queries, but also can answer advanced queries, like recommendation. Graph has been

widely used in recommendation for social networks [26], so methods can be referenced in workflow provenance graph. In this section, we explain three example recommendation scenarios that can be supported by our provenance model, as shown in Fig. 5.

Recommend collaborator based on design-time and running time. Fig. 5 (1) aims to answer Q2b query. Collaborators are not necessarily in the same domain. For example, a group of scientists in computer science are designing workflow for scientists in Earth science, so the latter check and give feedback to the former by running the workflow. If another group of Earth scientists is going to use similar tasks, the computer science group is very likely to be their collaborator. Furthermore, it is easy for us to rank our recommendations by their design-time performance.

*match (q)-[:ADD|EDIT|SAVE]->()-[:IsIncludedBy*0..5]->(:Workflow{name:' wfName ', version : ' version '})*
*<-[r:IsIncludedBy*0..5]-()<- [:ADD|EDIT|SAVE]-(p:Person{user:'user'}) return distinct(q)*

Recommend processor (sub-workflow) based on design-time and running time. As science and technology develop, there are numerous methods to accomplish a certain scientific task. However, different methods are good at different aspects so that they fit in different scenarios. For example, a method is suitable for running on large-scale datasets, while another method can achieve high accuracy on small datasets. Based on our running time and action history, we can recommend the most appropriate processor or workflow for a task under some specific scenario, as shown in Fig. 5(2).

match (:Workflow{name:' wfName ',version:'version'})<-[r:IsIncludedBy]-(n) *
unwind Labels(n) as l with distinct(l) as label, collect(n) as entity return label, entity

Recommend actions based on running time and action history. A scientific experiment is typically a trial and error process, so scientists will update their design after running. For example, after scientists running a workflow, they get output of geolocations, as shown in Fig. 5(3). It has been found in the action history that scientists will probably visualize the output points on the map, so we recommend the action "add processor 2Dvisualization" to them. Therefore, different outputs may lead to corresponding subsequent

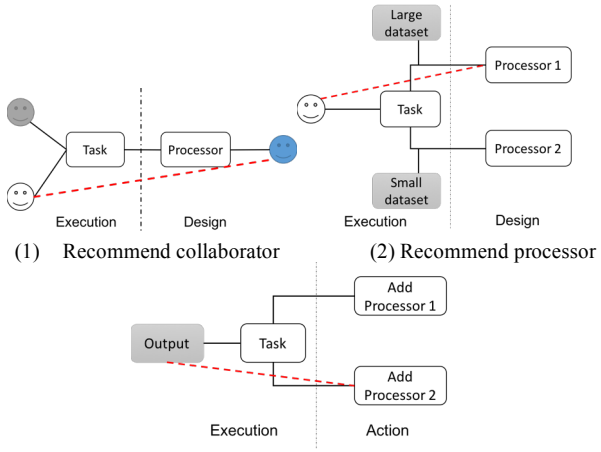


Fig. 5. Examples of three types of recommendation.

actions. The combination of running-time result and action history indicates frequent pattern of outputs and actions.

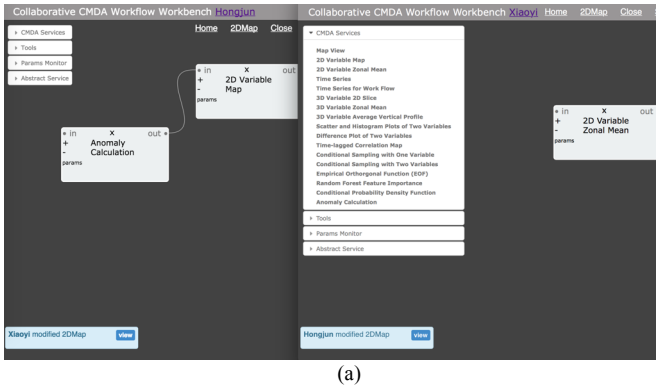
match (a:Action)-[:RELATESTO]->(w:Workflow)<-[r:IsIncludedBy]-(n) where id(a)="nodeId" *
unwind Labels(n) as l with distinct(l) as label, collect(distinct n) as entity return label, entity

VI. Implementations and Experiments

A. Prototype system

We have developed an online collaborative workbench system. It allows users to create workflows with multiple data processing web services and review workflow editing history tree. Users are allowed to collaborate on editing shared workflows, and create user groups for control the access to workflows.

The system contains four main components: workflow metadata management, workflow design and history, workflow customized query, and online collaboration. We adopted MongoDB to store user (group) information and Neo4j to manage workflow provenance, respectively. Fig. 6(a) is an example scenario of multi-user online collaboration, which supports both online and offline notifications. Assume Xiaoyi and Hongjun are two users collaborating in the same workflow project and editing at same time. Fig. 6(a) shows a scenario when Hongjun saved



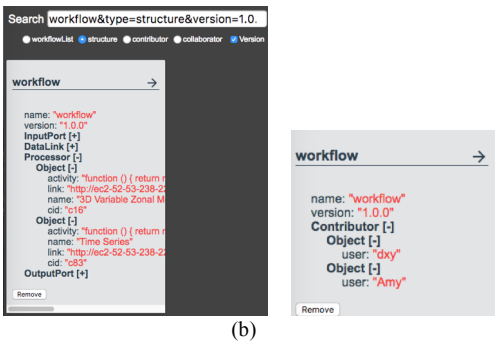


Fig. 6. Collaborative workflow design system and SQL-like query support.

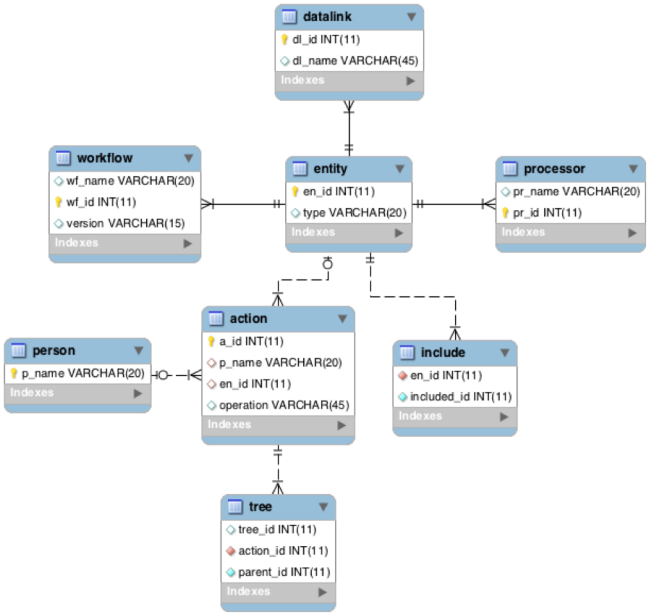


Fig. 7 Class diagram of Relational Database.

workflow first, Xiaoyi thus received the notification from Hongjun (right window). Afterwards, Xiaoyi also modified the workflow, and she decided to save her work before transmitting the notification signal. As a result, Hongjun received the notification from Xiaoyi as well (left window). Hence, their works were both saved. Fig. 6(b) indicates three customized query examples and results: (1) structure query is to retrieve workflow structure of a specific version; (2) contributor query is to get all contributors to a specific version of workflow; (3) collaborator query is to find who else also worked on the same workflow.

B. *Query Comparison Experiment*

To compare the efficiency of implementing our model in a graph database with a relational database, we designed data schema in SQL as shown in Fig. 7. First, we give the comparison of computational complexity of two query methods. We assume the number of query labels (types of nodes, such as workflow, processor, and action) in a graph database is n, and the average number of nodes for the same label is A. Accordingly, the number of queried tables in a relational database is n, and the average table size is A. If a query in a graph database is to find a path containing all n node labels, its time complexity will be $O(An)$. However, a query in a relational database has to join n tables, so its query complexity will be $O(A^n)$.

We also compared the performance using SQL and our SQL-like language to implement the three queries proposed in Section V(B). As shown below, writing queries in our SQL-like language can be simpler and more intuitive. For all kinds of queries, however, SQL must join multiple tables. Therefore, our language is not only easier for users to write, but also more efficient in execution. Screenshots of following queries result in our prototype system are shown in Fig. 6(b).

- Queries written in SQL:

Q1a:
```
use wf;
select processor.pr_name
from include,processor , workflow
where workflow.wf_name='workflow1' and
workflow.version='1.1' and workflow.wf_id = include.en_id
        and processor.pr_id = include.included_id
union
select datalink.dl_name
from include,datalink , workflow
where workflow.wf_name='workflow1' and
workflow.version='1.1' and workflow.wf_id = include.en_id
        and datalink.dl_id = include.included_id ;
```
Q2a:
```
Select p1. p_name, p2. p_name
From person as p1, person as p2, workflow,action
Where  workflow.wf_name='workflow1'
        and workflow.version='1.1'
        and p1.p_name!=p2.p_name
        and workflow.wf_id = action.en_id;
```
Q3a:
```
Select person.p_name
From person, action, include, workflow
Where  workflow.wf_name='workflow1'
        and workflow.version='1.1'
        and action.p_name = person.p_name
        and include.en_id = workflow.wf_id
        and action.en_id = include.included_id ;
```

C. *Loading time and Data size Experiments*

To verify the effectiveness of our model, we designed experiments in neo4j for data loading time, querying time and data storage size. We simulated data on three variables: number of people collaborated, number of actions implemented, and number of forks in the history tree. As the results in Table 1 show, a graph database can effectively answer all kinds of queries, especially recursive queries, which are not well-supported by a relational database.

Table 1. Simulation experiment in neo4j

#people	1	2	3
#Actions	4	8	12
#Forks	0	1	2
Loading time	30 ms	43 ms	71 ms
Structure query	3 ms	9 ms	12 ms
Contributor query	13 ms	17 ms	18 ms
Collaborator query	5 ms	9 ms	12 ms
History query	14 ms	20 ms	26 ms
Size	233 KB	243 KB	277 KB

VII. CONCLUSIONS

In this paper, we have demonstrated that the integration of design-time and run-time workflow provenance helps scientists trace their work and collaboration more effectively. Based on this finding, we have developed an integrated data model for workflow provenance management. Root in the cross-provenance model, we show that new applications can be developed, such as advanced query and cross-provenance recommendation.

In future work, we plan to further improve the efficiency of cross-provenance query based on the integrated graph.

Additionally, we also plan to utilize features extracted from provenance graph to improve recommendation accuracy.

ACKNOWLEDGMENT

This work is partially supported by National Science Foundation and National Aeronautics and Space Administration, under grants NSF ACI-1443069, NNX16AB22G, and NNX16AE15G. We appreciate Xiaozhe Wang, Hongyang Wang, and Hector Guo for their software development efforts.

REFERENCES

[1] Zhong, Y., Fan, Y., Huang, K., Tan, W. and Zhang, J. "Time-aware service recommendation for mashup creation," IEEE Transactions on Services Computing, 8(3), pp.356-368.

[2] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi, "Prospective and Retrospective Provenance Collection in Scientific Workflow Environments," in Proceedings of IEEE International Conference on Services Computing (SCC), 2010, pp. 449-456.

[3] Mattoso, Marta, and Boris Glavic, eds. "Provenance and Annotation of Data and Processes," 6th International Provenance and Annotation Workshop, IPAW 2016, McLean, VA, USA, June 7-8, 2016, Vol. 9672. Springer.

[4] J. Freire, C.T. Silva, S.P. Callahan, E. Santos, and C.E. Scheidegger, "Managing Rapidly-Evolving Scientific Workflows," Lecture Notes in Computer Science, vol. 4145/2006, 2006, 10–18.

[5] J. Freire, D. Koop, E. Santos, and C.T. Silva, "Provenance for Computational Tasks: A Survey," Computing in Science and Engineering (CSE), 10(3), 2008, pp. 11-21.

[6] Y. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in e-Science," SIGMOD Record, 34(3), 2005, pp. 31–36.

[7] W3C, An Overview of the PROV Family of Documents; Available from: https://www.w3.org/TR/prov-overview.

[8] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, and Y. Zhao, "Scientific Workflow Management and the Kepler System," Concurrency and Computation: Practice & Experience, 18(10), 2006, pp. 1039-1065.

[9] 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," Concurrency and Computation: Practice & Experience, 18(10), 2006, pp. 1067–1100.

[10] E.C. Withana, B. Plale, R. Barga, and N. Araujo, "Versioning for workflow evolution," Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 2010, pp. 756-765.

[11] P. Missier, , S.C. Dey, K. Belhajjame, V. Cuevas-Vicenttín, and B. Ludäscher. "D-PROV: Extending the PROV Provenance Model with Workflow Structure." In Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance (TaPP), 2013.

[12] "The Joint Task Force on Computing Curricula, Computing Curricula 2001," Journal of Educational Computing Research, 1(3), 2001, pp. 1-240.

[13] P. Groth, S. Miles, W. Fang, S.C. Wong, K.-P. Zauner, and L. Moreau, "Recording and Using Provenance in a Protein Compressibility Experiment," in Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing (HPDC), 2005, pp. 201-208.

[14] Y. Simmhan, B. Plale, and D. Gannon, "A Framework for Collecting Provenance in Data-Centric Scientific Workflows," in Proceedings of IEEE International Conference on Web Services (ICWS), 2006, pp. 427–436.

[15] The Open Provenance Model; Available from: http://openprovenance.org/.

[16] R. Bose and J. Frew, "Lineage Retrieval for Scientific Data Processing: A Survey," ACM Comput. Surv., 37(1) , 2005, pp. 1-28.

[17] V. Cuevas-Vicenttin, S. Dey, M. Wang, T. Song, and B. Ludäscher, "Modeling and querying scientific workflow provenance in the D-OPM," in Proceedings of High Performance Computing, Networking, Storage and Analysis, SC Companion, 2012, pp. 119-128.

[18] ProvONE: A PROV Extension Data Model for Scientific Workflow Provenance (2015); Available fron: http://vcvcomputing.com/provone/provone.html

[19] Garijo, Daniel, and Yolanda Gil. "Augmenting prov with plans in p-plan: scientific processes as linked data." CEUR Workshop Proceedings, 2012.

[20] O. Biton, S.C. Boulakia, S.B. Davidson, and C.S. Hara, "Querying and Managing Provenance through User Views in Scientific Workflows," in Proceedings of IEEE 24th International Conference on Data Engineering (ICDE), 2008, pp. 1072-1081.

[21] A. Chebotko, X. Fei, C. Lin, S. Lu, and F. Fotouhi, "Storing and Querying Scientific Workflow Provenance Metadata Using an RDBMS," in Proceedings of IEEE International Conference on e-Science and Grid Computing, 2007, pp. 611-618.

[22] O. Biton, S. Cohen-Boulakia, S.B. Davidson, and C.S. Hara, "Querying and Managing Provenance through User Views in Scientific Workflows," in Proceedings of IEEE 24th International Conference on Data Engineering, 2008, pp. 1072-1081.

[23] C. Scheidegger, D. Koop, E. Santos, H. Vo, S. Callahan, J. Freire, and C. Silva, "Tackling the Provenance Challenge One Layer at a Time," Concurrency and Computation: Practice and Experience, 20(5), 2008, pp. 473-483.

[24] Gadelha Jr, Luiz MR, Marta Mattoso, Michael Wilde, and Ian T. Foster. "Provenance Query Patterns for Many-Task Scientific Computing," In Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP), 2011.

[25] K. Anand, S. Bowers, and B. Ludäscher, "Techniques for Efficiently Querying Scientific Workflow Provenance Graphs," in Proceedings of EDBT, 2010, pp. 287-298.

[26] Konstas, Ioannis, Vassilios Stathopoulos, and Joemon M. Jose. "On Social Networks and Collaborative Recommendation," in Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2009, pp. 195-202.

[27] Zhang J, Bao Q, Duan X, Lu S, Xue L, Shi R, Tang P. "Collaborative Scientific Workflow Composition as a Service: An Infrastructure Supporting Collaborative Data Analytics Workflow Design and Management," IEEE 2nd International Conference on Collaboration and Internet Computing (CIC), 2016, pp. 219-228.

[28] Woodman, Simon, Hugo Hiden, and Paul Watson. "Workflow provenance: an analysis of long term storage costs." Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science. ACM, 2015.

[29] T. Heinis and G. Alonso, "Efficient Lineage Tracking for Scientific Workflows," in Proceedings of ACM International Conference on Management of Data (SIGMOD), 2008, pp. 1007-1018.

[30] A. Chapman, H.V. Jagadish, and P. Ramanan, "Efficient Provenance Storage," in Proceedings of ACM International Conference on Management of Data (SIGMOD), 2008, pp. 993-1006.