Storing and Discovering Critical Workflows from Log in Scientific Exploration

Qihong Shao
Arizona State University
qihong.shao@asu.edu

Michel Kinsy
Arizona State University
mkinsy@asu.edu

Yi Chen
Arizona State University
yi@asu.edu

1. Introduction

Workflows are widely used in various scientific fields, such as chemical physics, astronomy, environmental science and bioinformatics. A scientific workflow represents a set of experiments performed in an order that is consistent with certain constraints in order to achieve a scientific goal. We differentiate the design and the execution of a workflow. The design captures the conceptual model of the workflow, including the format of its input and output, the functionality of each experiment, as well as the logical constraints on experiment execution order. The execution of a workflow represents a sequence of the experiment runs, consisting of the actual input and output of each experiment, and the order of the execution that satisfies the specification in the design.

In a lot of scenarios, scientific discoveries are made as a result of a dynamic process, where scientists have a desired goal in mind without a clear workflow design. Toward the goal, they make a hypothesis and then try experiments for verification. Depending on whether the experiments succeed or not, they may refine the hypothesis or propose a new one, and then perform further experiments for validation. Usually many iterations of hypotheses and experiments may be needed and the whole process could take weeks or months. Once they succeed, it is important that the scientists and their colleagues can *reproduce* the scientific discoveries.

There are two challenges in reproducing the final result of a workflow execution in scientific exploration when the workflow design is not available. First, how should we document the log of all the experiments that have been performed such that the storage and access to the log can be effective and efficient? Database technology shows promising results for storing and querying workflow designs and executions, as exploited in [2, 4, 5]. However, these approaches require scientists to specify the designs of the workflows, based on which relational schemas are designed and corresponding SQL queries are generated, therefore do not fit for scientific exploration applications.

Second, how should we identify the experiments that are critical to reproduce the workflow results? In scientific exploration, not all the experiments in a workflow execution have necessarily contributed to the final result, since scientists may not have a good design before the execution and may keep proposing new hypotheses and verifying them during the execution till the success. Experiments that have parameter errors , execution errors, and/or design errors are not necessary to be repeated for reproducing the same result of the corresponding workflow execution. Identifying those experiments can reproduce the workflow results more efficiently.

Research efforts have been made to process workflow execution log in the absence of workflow design [1, 3]. However, the focus is on mining frequently occurring patterns in workflow execution log in order to infer the design. There is a lack of integrated support for storing and querying workflow executions. Furthermore, it is assumed that every experiment run in the workflow execution log is necessary in leading to the final success. Although this is often the case for business workflows, it may not always hold in scientific exploration where a scientist may keep proposing new hypotheses and verifying them during the workflow execution.

This paper aims at effectively reproducing the results of previous scientific workflow executions. We first identify the information that needs to be recorded in a workflow execution log, based on which we then determine the data flow among experiments. To effectively record a workflow execution log in a relational database management system (RDBMS) when the workflow design is absent, we propose a generic relational storage schema. Then techniques have been designed to automatically discover the minimal set of experiments that must be performed in order to reproduce a scientific result by posing appropriate SQL queries. Although such SQL queries can be evaluated using an off-the-shelf database system, we investigate the unique characteristics of the workflow log data and optimization techniques for evaluating such SQL queries efficiently.

N	T	I	O
C	T_1	1.1mmol dicarboxylic acid, 2.2mmol HCTU	99% Fc-1
S	T_2	methanol, 99% Fc-1	1mM methanol solution
P	T_3	gold wire	0.25μm probe
T	T_4	1mM methanol solution, 0.25μm probe	Test Enviroment1
M	T_5	Test Enviroment1, 0.2V bias	0.0mA
S	T_6	$HClO_4$, 99% Fc-1	1mM $HClO_4$ solution
T	T_7	1mM $HClO_4$ solution, 0.25μm probe	Test Enviroment2
M	T_8	Test Enviroment2, 0.2V bias	0.0mA
P	T_9	gold wire	0.15μm probe
T	T_{10}	1mM $HClO_4$ solution, 0.15μm probe	Test Enviroment3
M	T_{11}	Test Enviroment3, 0.2V bias	0.2mA
D	T_{12}	0.2mA	1000 transients
A	T_{13}	1000 transients	*analysis report*

Table 1. A sample execution log of a molecular electronics workflow

2. Recording Workflow Execution Log in RDBMS

The log of a workflow execution consists of a sequence of events, each of which records an experiment execution.

Definition 2.1: An event E of an experiment execution is described by a tuple (N, T, I, O), where N is the name of the experiment, T is the time when the experiment is performed [1], I and O are inputs and outputs of the experiment, respectively. We use $E.c$ to denote the c component of an event E, where c can be N, T, I or O. ■

Since an experiment can take multiple inputs and produce multiple outputs, both I and O denote sets. Each distinct input and output has a unique identifier (such as file name or material ID).

Example 2.1: Let us examine a workflow example in the area of molecular electronics (or Moletronics) [7] that investigates the electron transport properties of molecules. An execution log is shown in Table 1, consisting of a list of events in the order of their execution time. Each event is a run of one of the following experiments: Compound Synthesis (C), Solution Preparation (S), Probe Preparation (P), Testbed Setting (T), Current Monitoring (M), Data Collection (D), and Data Analysis (A).

In this workflow execution, the scientist first synthesized the ferrocene compounds, namely cysteamine-Fc-cyctermine (Fc-1) in the first experiment execution [C, T_1]. Then Fc-1 was dissolved in methanol to get a $\approx$ 1mM solution [S, T_2]. She also cut a gold wire to get a 0.25μm probe and checked the sharpness of its tip using microscope, cleaned the probe via a regular cleaning procedure [P, T_3]. Then a test bed with 0.2v bias supply was set up, using electrometer to monitor the current [T, T_4]. For a long time, there was no current observed, the experiments were considered as failed [M, T_5]. Next, the scientist decided to change to another solution $HClO_4$ [S, T_6], but used the same probe as the test bed [T, T_7]. However, there was still no current observed [M, T_8]. The scientist hypothesized that the probe was not sharp enough and had physically adsorbed molecules. Therefore she made a new probe with 0.15mm [P, T_9] and set up a new test bed [T, T_{10}]. This time, current was detected [M, T_{11}]. The scientist collected about 1000 transients [D, T_{12}], analyzed them using the Labview program, and finally achieved the goal [A, T_{13}]. ■

[1] We can record both the start time and finish time of an experiment run. For presentation simplicity, we use T to represent.

Before we present the proposed techniques, let us look at the existing approach [5] that leverages RDBMS for recording workflow execution log based on the workflow design. A relation is created for each experiment that is involved in the workflow design, which records the name, time stamp, each input and output parameter of the experiment. A tuple in the relation records an execution of the experiment. Then the data flow among experiment executions can be specified using SQL joins over the input and output attributes across different relations according to the workflow design specification.

Example 2.2: In the molecular electronics workflow, suppose that each of the experiments S and T takes two input parameters and produces a single output parameter. The relational schema for recording their experiment runs are as follows:
T{Name, Time, I1, I2, O } ,
S{Name, Time, I1, I2, O } .

Furthermore, the workflow design states that the second input parameter of an experiment T is obtained from the output of an experiment S, then we can specify an inclusion dependency: for any value of attribute I_2 in relation T, there exists a value of attribute O in relation S. Therefore we can easily find out the data flow by joining the corresponding attributes in relation S and T. ■

However, if the workflow design is not available, although we can still create one relation for each possible experiment according to its input and output parameters, the relationship between the inputs and outputs of two experiments is unclear before the experiment execution. Furthermore, the connection between two experiments can vary for different workflows. Therefore, it is hard to automatically generate SQL queries to determine the data flow among experiment executions. In Example 2.2, without the knowledge of the workflow design, it is not clear which attributes in which relations should be joined together in SQL queries in order to determine the data flow and construct a workflow execution from individual experiment .

In order to effectively record workflow execution log and reconstruct data flow in the absence of workflow design, we propose a generic relational schema to record workflow execution log:
Exp{expID, wfID, Name, Time},

ExpIn{expID, Input},
ExpOut{expID, Output}.

The key attributes of each relation are underlined. For the *Exp* relation, an alternative key is *(Name, Time, wfID)*. As will be shown in Section 4 shortly, SQL queries on this schema can be easily generated to build the workflow data flow from individual experiment execution.

Example 2.3: Given the above relational schema, the workflow execution log represented in Table 1 is stored in the relations in Figure 1. ∎

3. Identifying Critical Workflows from Log

The execution of a workflow can be conceptually represented as a graph G_I, named as initial graph.

Definition 3.1: An *initial graph* $G_I = (V, E, S)$ of a workflow execution log L is a directed acyclic graph constructed as follows. Each node v in G_I represents a distinct event in L: $f(v)$, where f is the function that maps a node in V to the corresponding event in L. All these nodes and an initial node v_0 consist of the node set V. For any two nodes v_i, $v_j \in V$, there is an edge from v_i to v_j in G_I if and only if $\exists d \in f(v_i).O$, $d \in f(v_j).I$. There is also an edge from the initial node v_0 to each node v_i, if $\exists d \in f(v_i).I$, such that there does not exist $v_j \in V$, $d \in f(v_j).O$. The edge set E consists of all such edges. $S \subseteq V$ denotes the successful states of the workflow execution. ∎

Note that G_I is an acyclic graph since each execution of an experiment is represented as a distinct node. The direction of the edges in G_I represent the data flow direction between two consecutive experiments.

Example 3.1: Continuing our running example, the initial graph of the workflow execution log in Example 2.1 is presented in Figure 2. For presentation conciseness, we use N_i to represent the event with name N that occurred at time T_i. For instance, S_6 denotes the execution of the experiment of name S at time T_6 in the log. Also we introduce node I to represent the initial node. ∎

Consider the initial graph representation of the execution log, the nodes on the paths from the initial node to a node in the successful states S are considered to be critical since their outputs contribute to the final result of the workflow execution. Those nodes and the edges connecting them consist of the critical workflow graph.

Definition 3.2: A *critical graph* $G_C = (V', E', S)$ is a subgraph of an initial graph $G_I = (V, E, S)$. $V' \subseteq V$, such that every node $v' \in V'$ has a directed path from v' to a node $s \in S$. $E' \subseteq E$, such that for every edge $e' \in E'$, the nodes that e' connects to are in V'. ∎

Example 3.2: The critical graph G_C for the initial graph

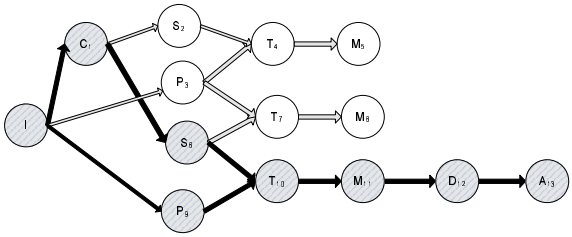

Figure 2. Initial graph G_I; Critical graph G_C consisting of only highlighted nodes and edges

G_I in Figure 2 consists of the nodes that are highlighted in gray color as well as the edges connecting them. ∎

4. Retrieving Critical Workflow Graphs Using SQL Queries

Given the relational schema presented in Section 2 for recording workflow execution log, the problem of identifying critical workflow graphs can be solved using SQL queries.

We first build a view *ExpLogic{expID, wfID, Name, Time, Input, Output}*, which represents experiment executions with both input and output information using a natural join on relations *Exp*, *ExpIn* and *ExpOut*. Since an experiment can take several inputs and produce several outputs, it may correspond to multiple tuples in *ExpLogic*. According to Definition 3.1 and 3.2, there is an edge in the initial or critical workflow graph from the node representing event E_1 to the node representing event E_2 if and only if the output set of E_1 has a non-empty intersection with the input set of E_2, that is: $E_1.I \cap E_2.O \neq \emptyset$. The data flow can be identified by checking whether there exists a tuple t_1 in *ExpLogic* corresponding to E_1, and a tuple t_2 corresponding to E_2, such that t_1.*Input* = t_2.*Output*.

The view of the critical workflow graph can be constructed using recursive SQL queries. Initially, the tuples in *ExpLogic* whose *Output* attribute values match the desired final workflow output are retrieved and composed of a view *ExpCritical*. Then the query recursively retrieves tuples whose *Output* values match the *Input* of a tuple in the current *ExpCritical* view through joins. That is, nodes that are connected to a node in the current critical graph become part of the expanded critical graph. The result of *ExpCritical* is finalized when no more tuples can be added. The initial workflow graph can be constructed similarly. Due to space limitation, we refer readers to [6] for more details.

The SQL queries for constructing critical workflow graph can be evaluated using an off-the-shelf relational database. We observe a special feature in the workflow log

expID	Name	Time	wfID
1	C	T_1	1
2	S	T_2	1
3	P	T_3	1
4	T	T_4	1
5	M	T_5	1
6	S	T_6	1
7	T	T_7	1
8	M	T_8	1
9	P	T_9	1
10	T	T_{10}	1
11	M	T_{11}	1
12	D	T_{12}	1
13	A	T_{13}	1
14	C	T_{14}	2
...	...	...	...

Table (1) Exp

expID	Input
1	1.1mmol dicarboxylic acid
1	2.2mmol HCTU
2	99% Fc-1
2	methanol
3	gold wire
4	1mM methanol solution
4	0.25 μm probe
5	Test Enviroment1
5	0.2V bias
6	99% Fc-1
6	HClO4
7	1mM HClO4 solution
7	0.25 μm probe
8	Test Enviroment2
8	0.2V bias
9	gold wire
10	1mM HClO4 solution
10	0.15 μm probe
11	Test Enviroment3
11	0.2V bias
12	0.2mA
13	1000 transients
14	...

Table (2) ExpIn

expID	Output
1	99% Fc-1
2	1mM methanol solution
3	0.25 μm probe
4	Test Enviroment1
5	0.0mA
6	1mM HClO4 solution
7	Test Enviroment2
8	0.0mA
9	0.15 μm probe
10	Test Enviroment3
11	0.2mA
12	1000 transients
13	...

Table (3) ExpOut

Figure 1. A generic solution for recording workflow execution log in RDBMS

data, tuples representing experiments are recorded into the database in the order of experiment execution time. We are investigating efficient implementation of joins among *ExpLogic*, *ExpCritical* and *ExpInitial* relations by leveraging this feature, as discussed in an extended version of the paper [6].

5. Conclusions and Future Work

In this paper, we have examined the problem of effectively reproducing the results of exploratory scientific workflows. We have proposed a generic relational schema for leveraging RDBMS to record the workflow execution log which does not require workflow design information and therefore is suitable for scientific exploration applications. Not all the experiments performed in a workflow execution have necessarily contributed to the final desired results due to possible wrong parameters, abnormal termination, and/or design errors. We have designed SQL queries to identify the critical experiments that contribute to the successful states and the logical constraints on their execution order from the log database, such that workflow execution can be reproduced efficiently. We are designing and implementing optimization techniques for efficiently evaluating such SQL queries. In the future, we plan to exploit a repository of workflow execution log, and present relevant existing workflows to assist scientific users for constructing new workflows by querying the database effectively.

6 Acknowledgments

This research is partially supported by NSF IIS-0612273.

References

[1] Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In *EDBT*, pages 469–483, 1998.

[2] I-Min A. Chen and Victor M. Markowitz. Modeling scientific experiments with an object data model. In *International Conference on Data Engineering*, pages 391–400, 1995.

[3] Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Trans. Softw. Eng. Methodol.*, 7(3):215–249, 1998.

[4] David T. Liu and Michael J. Franklin. The design of griddb: A data-centric overlay for the scientific grid. In *VLDB*, pages 600–611, 2004.

[5] Srinath Shankar, Ameet Kini, David J. DeWitt, and Jeffrey F. Naughton. Integrating databases and workflow systems. *SIGMOD Record*, 34(3):5–11, 2005.

[6] Qihong Shao, Michel Kinsy, and Yi Chen. Storing and efficiently querying critical workflows from log in scientific exploration. Technical Report TR-07-005, Arizona State University, April 2007. http://wsdb.eas.asu.edu/publications/TR-07-005.pdf.

[7] X.Y. Xiao, D. Brune, J. He, S.M Lindsay, and C. B Gorman; N.J. Tao. Redox-gated electron transport in electrically wired ferrocene molecules. *Chemical Physics Lett.*, 2006.