User-friendly Provenance Management in Swift/T

Jennifer A. Steffens
Drake University
jennifer.steffens@drake.edu

Justin M. Wozniak
Argonne National Laboratory
wozniak@mcs.anl.gov

ABSTRACT

In scientific computing, understanding the origins and derivation of data is crucial. Provenance models aim to provide a means of capturing this in an efficient and effective manner. For the Swift/T language, the current provenance handling system requires improvement. In this proposal, I discuss the development of a new Swift/T provenance model based on the Swift/K provenance model, which would store provenance logs in a user-accessible database for future analysis.

BACKGROUND INFORMATION

The Swift system is designed for optimizing the execution of scientific computational experiments by performing independent tasks in parallel. Swift/T is an implementation of this language, in which the script is translated into a Turbine and MPI program. It is also capable of calling native code functions and executing scripts in embedded interpreters, allowing it to be compatible with languages such as C, C++, Python, and Fortran.

Due to the massive scale of the data resulting from the execution of large workflows, manual analysis of results is impractical. Instead, provenance systems can be utilized, as they are designed to gather details about the execution of the program, which in turn makes it easier to analyze and reproduce the outputs of computational experiments.

Swift logs information about each execution, and in the Swift/K implementation, the predecessor to Swift/T, a provenance model was proposed based on the Open Provenance Model (OPM) and the Virtual Data System (VDS) that exported these logs to a database, which could be queried after program termination for details concerning the results and execution of the workflows.

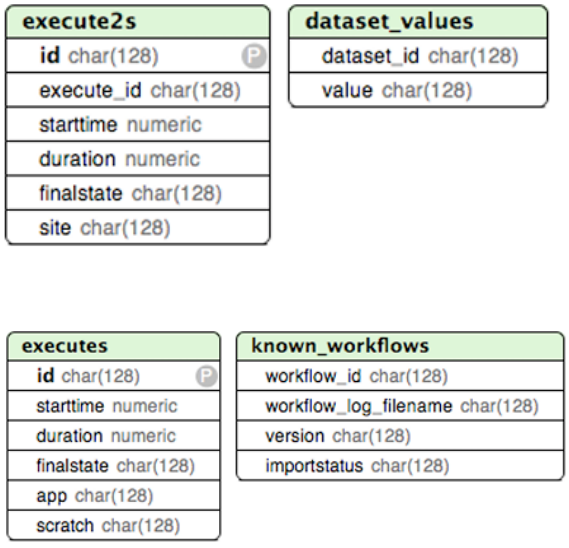
Problem Statement

Swift/T presently logs various information about a workflow as it executes, and displays this information to the console upon completion using a Tcl program. This information is deleted immediately after it is displayed, and is not displayed at all if the program encounters an error and terminates prematurely. In addition, the information displays in a single list, making finding specific details tedious, and is difficult for one without computational knowledge to decode.

Related Work

Open Provenance Model – OPM is an open source model that was developed as a result of the first provenance challenge. Its main objective is to allow provenance information to be exchanged between systems. It defines entities *artifact*, *process,* and *agent,* and relationships between them in order to assert casual dependencies. Its design is general, and needs to be modified to fit the specificity of Swift's provenance process.

The Swift/K Provenance Model – The model for the previous generation of the Swift language is based on OPM. After the execution of parallel scripts that specify many-task computations, this model extracts provenance information from the logs that Swift/K generates and stores it in a relational database [Figure 1].



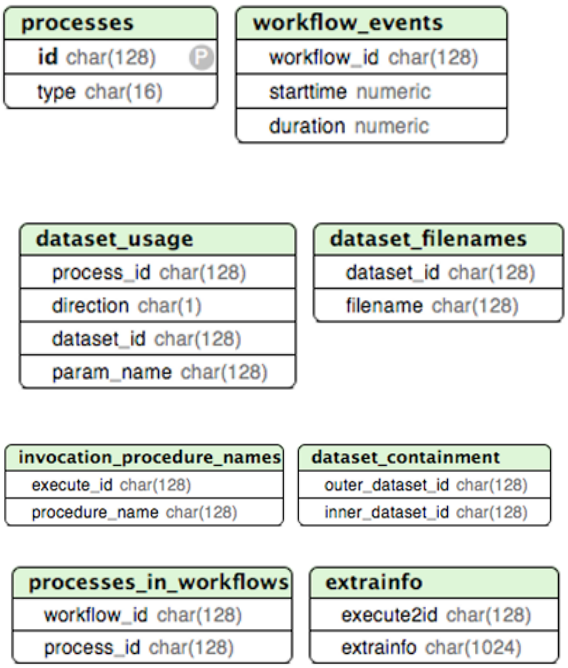


Figure 1: Swift/K's provenance relations (2010)

The developers also identified patterns for application independent queries and stored procedures for the patterns that were more difficult to express with relational database queries, thereby addressing some of the weaknesses of using a relational database. Since it was specifically made for Swift/K, this model requires modification of its schema to fit Swift/T. In addition, this model extracts information from log files only after a execution terminates. This makes it hard to track a run's progress.

PROPOSED SOLUTION

I propose developing a Swift/T provenance model based on the Swift/K model that efficiently organizes and saves information to a database in real-time and provides users a means by which to track progress as a workflow executes. I will be using Tcl integrated with SQLite to create and query the database. By examining Swift/T's current logs, I will be able to determine which relations and attributes are needed to effectively track and record the workflow.

From the found Swift/K query patterns, I will be able to develop functions that hold the complex procedures required, thereby increasing the ease of using the system. In addition, I aim use Tcl/Tk to create a graphical user interface (GUI) that can display a program's details during its execution without needing user queries, and can also function as an environment in which users can connect to the database directly to make specific queries.

Evaluation and Deliverables

Once I have a schema set up and functions utilized, I will monitor the database as it handles different types of workflows. I will experiment with tracing back certain outputs to test the effectiveness of my schema, and will modify the relations and attributes as I see fit based on the results in order to streamline the provenance process [Figure 2]. This project will be successful if the database produced retains its functionality across multiple different types of workflows.

Week	Goal
3	Complete Tcl program to build and fill database
4	Create functions for the query patterns
5	Test and modify schema and functions
6	Create GUI
7	Integrate software into Swift/T
8	Finishing touches, address any issues
9	Prepare report

Figure 2: My work schedule for the program

At the conclusion of this project, I will have code that can be integrated into the Swift/T source code and will handle the database. In addition, I will have a program that runs the provenance GUI.

CONCLUSION

Swift/T is a language designed to be useful in all fields, not exclusively computational science. It can be used to model epidemics, analyze a protein's structure, or any other task that requires efficient large-scale parallel processing. In order to allow simple data derivation tracking in other fields, I am developing a user-friendly Swift/T provenance model that will store information in a relational database, hold query functions, and include a GUI for easy database access.

By working on this summer project, I will be able to extend my knowledge of relational databases, acquire new languages like Tcl and Swift/T, understand how a parallel processing program handles data, learn the process through which programming languages are developed, learn GUI programming, and study the concept of data provenance.

In addition to technical skills, I will gain the experience of working in research full time, and having to collaborate and learn from other people, including my mentor, my peers, and the authors of the papers that I can get in contact with, which will better prepare me for graduate school.

REFERENCES

Armstrong, Timothy G., Justin M. Wozniak, Michael Wilde, and Ian T. Foster. "Compiler techniques for massively scalable implicit task parallelism." In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 299-310. IEEE Press, 2014.

Gadelha, L. M., B. Clifford, M. Mattoso, M. Wilde, and I. Foster. *Provenance management in Swift with implementation details*. No. ANL/MCS-TM-311. Argonne National Laboratory (ANL), 2011.

Gadelha Jr, Luiz MR, Marta Mattoso, Michael Wilde, and Ian T. Foster. "Provenance Query Patterns for Many-Task Scientific Computing." In *TaPP*. 2011.

Gadelha Jr, Luiz MR, Michael Wilde, Marta Mattoso, and Ian Foster. "MTCProv: a practical provenance query framework for many-task scientific computing." *Distributed and Parallel Databases* 30, no. 5-6 (2012): 351-370.

Moreau, Luc, Juliana Freire, Joe Futrelle, Jim Myers, and Patrick Paulson. "Governance of the Open Provenance Model." *URL http://twiki.ipaw.info/ pub/OPM/WebHome/governance. pdf* (2009).

Moreau, Luc, Beth Plale, Simon Miles, Carole Goble, Paolo Missier, Roger Barga, Yogesh Simmhan et al. "The open provenance model (v1. 01)." (2008): 20.

Wozniak, Justin M., Timothy G. Armstrong, Michael Wilde, Daniel S. Katz, Ewing Lusk, and Ian T. Foster. "Swift/T: large-scale application composition via distributed-memory dataflow processing." In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pp. 95-102. IEEE, 2013.

Wozniak, Justin M., Michael Wilde, and Ian T. Foster. "Language features for scalable distributed-memory dataflow computing." In *Data-Flow Execution Models for Extreme Scale Computing (DFM), 2014 Fourth Workshop on*, pp. 50-53. IEEE, 2014.

Zhao, Dongfang, Chen Shou, Tania Malik, and Ioan Raicu. "Distributed data provenance for large-scale data-intensive computing." In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pp. 1-8. IEEE, 2013.