DISCLAIMER

A Noisy 10GB Provenance Database

You-Wei Cheah[1], Beth Plale[1], Joey Kendall-Morwick[1], David Leake[1], Lavanya Ramakrishnan[2]

[1]School of Informatics and Computing, Indiana University, Bloomington IN
[2] Lawrence Berkeley National Lab, Berkeley, CA
{yocheah, plale, jmorwick, leake}@cs.indiana.edu, lramakrishnan@lbl.gov

Abstract. Provenance of scientific data is a key piece of the metadata record for the data's ongoing discovery and reuse. Provenance collection systems capture provenance on the fly, however, the protocol between application and provenance tool may not be reliable. Consequently, the provenance record can be partial, partitioned, and simply inaccurate. We use a workflow emulator that models faults to construct a large 10GB database of provenance that we know is noisy (that is, has errors). We discuss the process of generating the provenance database, and show early results on the kinds of provenance analysis enabled by the large provenance.

Keywords: Data provenance, scientific workflows, provenance quality, case-based reasoning.

1 Introduction

Data provenance provides the lineage or history of how data is generated. Provenance information is valuable in scientific datasets because it may be the only source of comprehensive information about how an e-Science data product was arrived at. Because the scientific workflow development process involves scientists refining their workflows repeatedly over time, provenance can also help scientists track their decisions and enhance the process of finding the optimum workflow. Moreover, provenance supports experiment reproducibility and reuse of scientific data and can contribute to assessments of the quality of a data set [19]. With the increase in awareness of the importance of preserving society's investment in data driven research, the need for useful data provenance has become increasingly critical.

Research on provenance in scientific workflows has focused on provenance capture and management, resulting in systems such as Karma [18] and provenance support in workflow tools such as Kepler [11] and Pegasus [7]. In order to capture provenance, workflow engines must be instrumented or logs mined. Provenance traces are stored and managed using an internal provenance data model, often with interoperability support using the Open Provenance Model (OPM) [13].

Provenance is often not complete. The protocol between application and the provenance storage can be unreliable [4]. Additionally, the entire category of semi-structured workflows assumes there are gaps in the provenance record. Semi-structured workflows in e-Science encompass automated and non-automate

components where the specification is not known in advance.

In order to study scalable analysis techniques that are resilient to errors in provenance data, we built a 10GB database of provenance data with known failure patterns. In this paper, we define the methodology behind the database's construction. The database is populated from a workload of workflows that are modeled based on real workflows. The workflows making up the workload originate in a number of scientific domains. We emulate different workflow execution scenarios by controlled injection of failures and message drops during workflow execution. We examine the resulting distribution and include performance evaluations for the generation process of the database. As the larger research goal guiding this effort is analysis techniques for provenance use that run at scale and are resilient to failures, we discuss early work on two analysis approaches, one a graph analysis approach to detecting inferior workflow runs, and one that uses reasoning techniques to repair provenance graphs.

The remainder of this paper is organized as follows. Section 2 discusses related work and Section 3 identifies requirements for generating the gigabyte provenance database. Section 4 discusses the system components used to generate the database; Section 5 details the workflow workload. In Section 6, we discuss our methodology. Section 7 evaluates the performance, and Section 8 discusses provenance analysis enabled by the research. Section 9 concludes the paper and discusses future work.

2 Related Work

Provenance research falls primarily into main categories: 1.) business provenance, 2.) provenance capture that is tightly coupled to a workflow system, 3.) database provenance, and 4.) provenance capture in semi-structured e-Science environments. Over the years, multiple surveys [4, 20] have been conducted and have mapped out provenance systems in these categories. An example of business provenance involves lineage tracing in data warehousing systems [3]. For the other three categories in provenance research, a few example systems are Kepler [11] and Pegasus [7] (Category 2), Trio [22] (Category 3), and ES3 [5] (Category 4). Systems such as Karma [18] involve provenance research in two categories (Category 2 and 4). These systems provide a source for realistic provenance data; however, these systems do not provide a controlled provenance generation environment and do not necessarily contain provenance with failures. This is the missing gap that this paper addresses.

Many synthetic workloads have been developed and used over the years, several in the area of distributed systems [2, 12, 21]. Similarly, a number of workloads [1, 14] have been generated and used in networks research. These workloads were developed for performance evaluations, and for benchmarking purposes in their respective areas. However, none of these workloads attempt to model failures. To the best of our knowledge, no workloads have been developed specifically for the purpose of provenance research. With the creation of a noisy 10GB provenance database that models failures of provenance notifications, we present a synthetic database that reflects the needs of provenance research.

3 Provenance Database Requirements

For a provenance database to be useful for study, several requirements must be met:

Large scale. The database should consist of a significant number of provenance records to allow research to be done at scale.

Diversity. The provenance in the database should be drawn from workflows that are varied, such as those originating from different scientific domains and which have different characteristics in terms of size, breadth, and length.

Realism. The composition of workflows used to generate the provenance should have different availability and failure characteristics that are reflective of workflows that occur in the real world.

Using the WORKEM [16] workflow emulator to generate provenance, the six major workflows developed as part of the emulator, and the failure model built into WORKEM, we have achieved scale, diversity, and realism in the 10GB provenance database.

4 System Components

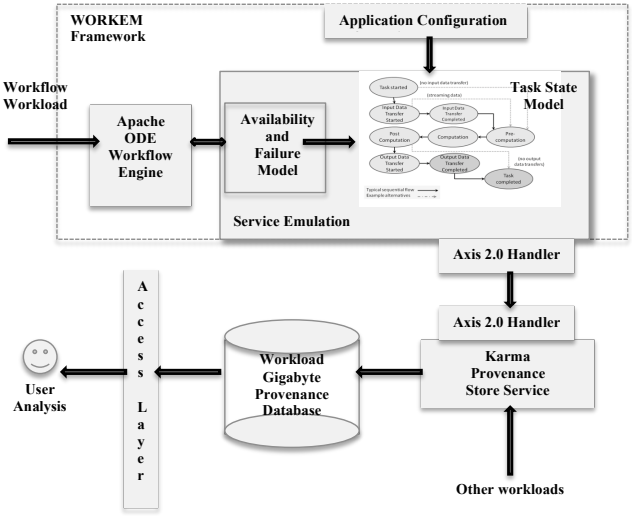

Fig. 1. Workload Gigabyte Provenance Database generation framework.

The two components used in the creation of the provenance database are WORKEM and Karma version 3.0. Figure 1 gives an overview of the system framework used to populate the workload gigabyte provenance database.

WORKEM is an emulation framework that emulates workflow execution [16]. It consists of an application service emulation layer that is built on top of a workflow engine, Apache ODE, and a task state model. Workflows are coded as BPEL workflow scripts and workflow notifications are generated through a generic service

that models task execution as a finite state machine. An availability and failure model is built into WORKEM enabling the modeling of different workflow scenarios. This model allows the user to configure the probability of dropping messages or failure for any node in WORKEMs task state model.

The current implementation of WORKEM is deployed with a suite of workflows based on a workflows survey [15]. The workflows are modeled using Xbaya [17]. We use the existing suite of workflows in the population of the database. These workflows will be further described in Section 5.

In Figure 1, workflow scripts are loaded into the Apache ODE workflow engine. The workflow is orchestrated by ODE, which instead of calling out to a real task, calls an emulated task that has been configured to have the black-box behavior of a real workflow node. Prior to the workflows execution, it passes through a failure model to see if the node should be "executed" at all, or if it should send erroneous information. The workflow task sends provenance notifications to Karma through Axis 2. Upon receipt the provenance notifications are integrated into the database.

Karma version 3.0 [18] is a provenance collection and management system. In this study, it is used to consolidate and store notifications generated by WORKEM. Karma is a versatile provenance system in that it accepts provenance in a number of ways. Karma is able to listen on a message bus or receive messages directly through a web service interface. Asynchronous threads process provenance notifications to extract provenance information and store the information to a relational Karma database that is OPM compatible. We have instrumented WORKEM with an Axis 2.0 handler to facilitate a direct transfer of notifications from WORKEM to Karma.

For each state in the WORKEM task state model, a message containing activity information is passed to Karma and translated into Karma's information model. Karma in turn populates the workload gigabyte provenance database using translated raw workflow notifications. Messages are represented using service invocations, data transfers, response status messages and computational messages.

The access layer shown in Figure 1, is an access interface to the provenance store. Currently, Karma supports a number of query API calls to ease the retrieval of provenance information. However, multiple access layers may be implemented to serve different purposes.

5 Workflow Workload

The provenance database is generated from the following six workflows, namely:

i. LEAD North American Mesoscale (NAM) initialized forecast workflow
ii. SCOOP ADCIRC Workflow
iii. NCFS Workflow
iv. Gene2Life Workflow
v. Animation Workflow
vi. MotifNetwork Workflow

These workflows are pseudo-realistic, in the sense that they are modeled after real life workflows [15] using WORKEMs task state model. The LEAD NAM, SCOOP and NCFS are weather and ocean modeling workflows, Gene2Life and MOTIF are

bioinformatics and biomedical workflows, and the Animation workflow carries out computer animation rendering. Some of the workflows are small, having few nodes and edges, while others like Motif have a few hundred nodes and edges. The characteristics are summarized in Table 1.

Table 1. Overview of Workflow Structure

Workflow Name	Number of Nodes (Tasks)	Number of Edges	Maximum Width
LEAD NAM	6	11	3
NCFS	7	19	2
SCOOP	6	10	5
Gene2Life	8	15	2
Animation	22	42	20
Motif	138	275	135

6 Methodology

We model failures in two specific ways a) task failures where a node in a workflow does not complete successfully b) a task completes but the notification is not successfully transmitted. These failure rates are modeled using uniform distributions in the emulator to determine if a particular invocation must fail or drop a notification. To generate the database, each of the six workflow types is run 2000 times per failure mode, with the failure modes as follows:

i. No failures and dropped notifications (success case)
ii. 1% failure rate
iii. 1% dropped notification rate
iv. 1% failure rate and 1% dropped notification rate

Specifically, WORKEM generates notifications based on a task state model using workflows coded as BPEL workflow scripts. A total of 9 states are present within the task state model. These states represent different workflow execution states and can be categorized into status notifications; computation notifications and data transfer notifications. The failure and dropped notification rates were configured for all states in WORKEMs task state model. These 4 population cases were determined based on preliminary testing, which displayed a good number of workflows with different characteristics. Using these configurations, we were able to achieve a wide variety of workflow execution traces by using the above configurations.

For each population case, we configured WORKEM to generate workflows using 10 threads in parallel, with each thread responsible for generating 200 workflows for a total of 2000 workflows. This process was repeated across a total of 6 workflow types with a goal of generating a total of 48,000 workflows.

WORKEM generated roughly 48,000 workflows with various failures and dropped messages. The total number of workflows differs slightly from the intended number for a few reasons. For the SCOOP workflow, we encountered a single failure in Apache ODE during generation through WORKEM. For the Animation workflows,

50 workflows were removed from the database due to an error during configuration. The causes of the 36 missing Motif workflows remain unknown.

As shown in Figure 2, the distribution is surprisingly dissimilar. Even though the generation settings for WORKEM were identical across workflows, we observe that WORKEMs failure model does not result in the same uniform distribution across different workflows since the configuration for failure rates are per task in the workflow. This is evident through the Animation and Motif workflows. As seen in the success category of Figure 2, only 2000 Motif workflows result without any failures. All of these workflows originate from the workflow run that was configured without any failures. Comparatively, for the failed case, we observe a total of 2430 workflows. Similarly, Animation workflows only have 2197 workflows without failures, whereas it has 2907 workflows with dropped and failed characteristics. Both Animation and Motif workflows that do not have failures or dropped messages are approximately half of what the smaller workflows exhibit, that confirms that the larger a workflow, the higher the failure rate and dropped messages rate.

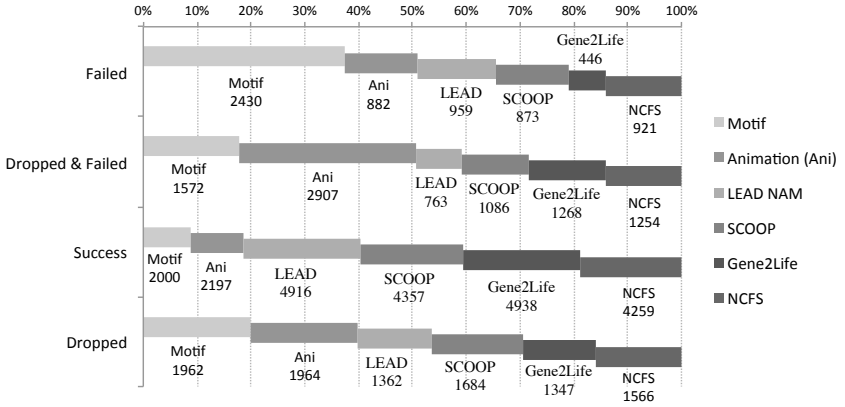

Fig. 2. Distribution of workflows by population cases.

The smaller workflows appear to have the same distribution amongst each other. As seen in Figure 3, about 55-60% of these workflows have no failures and dropped messages, while workflows with dropped messages are approximately 20% and workflows with failures or dropped messages accounting for the remaining 20-25%. The larger Motif and Animation workflows have a different distribution. Approximately 50% of these workflows generated appear to be failed workflows, while the other half is split between workflows that have dropped messages and successful workflows.

7 Performance Evaluation

We examined performance of the provenance database generation process to better understand the complexities involved in generation. We use as our testbed a Dell PowerEdge 6950, quad dual-core AMD Opteron 2.4GHz with 16GB of RAM running

Red Hat Enterprise Linux version 2.6.9-89.29.1.ELsmp. Both WORKEM and Karma were run on this machine. MySQL server v5.0.41 is the database system and it uses the machine's local disk. As populating the database took considerable time, it was carried out while other work was going on the server.

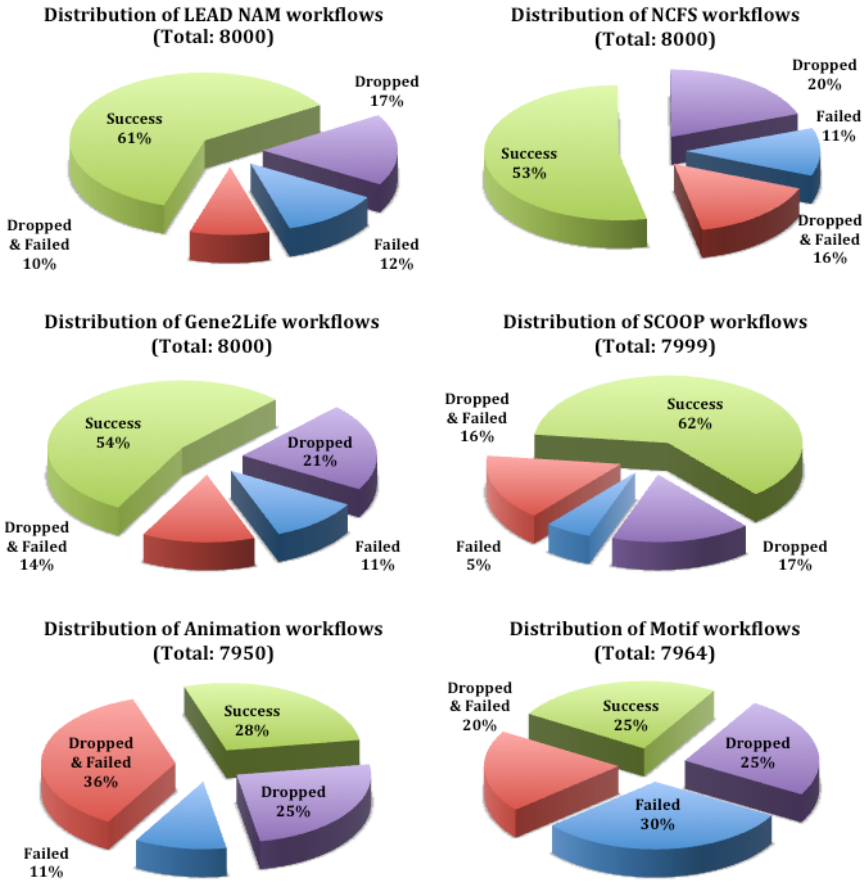

Fig. 3. Distribution of workflows by workflow types.

Analysis. The average population time per workflow for the different population cases discussed in Section 6 is presented in Table 2. We note a number of interesting observations. For all workflows, the average population time per workflow is the largest for the population case with dropped notifications. The LEAD NAM workflow is the sole workflow that does not exhibit this, but even then the average population time per workflow is fairly close to that of the case without failures or dropped messages. We also observe that the population case without failures or dropped notifications is significantly faster when compared to the population case with dropped notifications.

The larger the workflows, the longer the average population time per workflow for all population cases. This is evident in larger workflows such as Motif and Animation.

In these workflows, population cases that involve failures have the lowest average population time, indicating that most of these failures occur earlier in the workflow. This is especially evident in the Motif workflows. For Gene2Life and NCFS workflows, we observe that the population case with no failures or dropped notifications has a substantially lower average time than the population cases with failure rates or dropped notification rates.

Table 2. Average population time per workflow organized based on population cases.

Workflow Runs / Workflow Types	Success case (sec.)	1% failure rate (sec.)	1% dropped notification rate (sec.)	1% failure rate & 1% dropped notification rate (sec.)
Animation	28.2	17.3	35.3	21.3
Gene2Life	7.4	21.8	26.9	20.8
LEAD NAM	8.6	6.5	8.5	6.3
Motif	198.9	29.8	216.4	41.4
NCFS	7.2	21.7	23.1	16.8
SCOOP	19.1	21.4	24.0	23.2

Workflow Population Characteristics. We further examine characteristics of population time for the various workloads. We plot population times of each workflow run (y-axis) based on the start time for each workflow (x-axis). Figure 4 shows the database being populated with workflow provenance in a well-behaved manner. A couple of the workflows (NCFS and Gene2Life) showed a sudden decrease in population time by 75% around half way through the population cycle. We do not show this graph as it is likely due to background activity on the machine. The largest workflow, Motif, shows a partitioning in population time for the failure cases that reflects completion times shown in Table 2 for Motif (Figure 5). The 1% dropped notification rate averages 216 seconds while the 1% failure rate combined with the failure+dropped case (rightmost column of Table 2), averages 35 seconds.

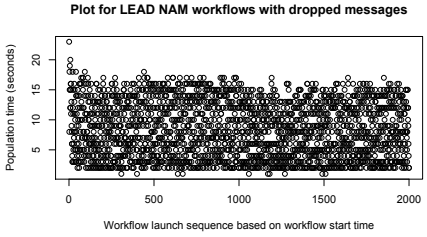

Fig. 4. Plot of workflows with uniform distributions in population timings.

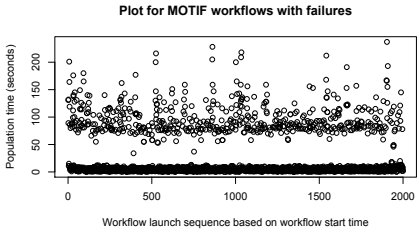

Fig. 5. Partitioning in population timings for Motif workflows that involve failures.

Size of Database. The total size of the workload gigabyte provenance database dump using Karma version 3.0 is 10.64 Gigabytes. This is a sizable database that takes approximately an hour and 5 minutes on average to import into MySQL on our experimental quad dual-core server.

8 Towards Large-Scale Provenance Analysis

The 10GB provenance database was developed to serve as a test platform for research into analysis algorithms that run at scale and are resilient to failures. Here we discuss two ongoing efforts.

Provenance Quality Assessment. Provenance, as we have already pointed out, can be messy. Provenance messages may be dropped, messages can be incomplete, which could occur when the application scope at a point of notification generation is more restricted than anticipated, or execution of the application (or workflow) can simply fail. We are examining fast statistical approaches that operate over large volumes of data to zero in on suspicious provenance records. Provenance goodness is determined by constructing the best possible provenance graph for an execution based on the captured provenance record, then assessing the goodness of the resulting graph by looking at the partitions in a provenance graph. A provenance graph can be modeled as $P_G = \{V, E\}$, where V is a collection of vertices that are linked by one or more directed edges, E.

The approach we use is to construct a provenance graph from nothing (no guiding workflow template) based only on the captured provenance. A current assumption of the approach is that all notifications contain the correct ID for the workflow execution instance to which they belong. WORKEM supports this assumption. While simplifying the problem, this approach still may yield disconnected components. The query of a graph using a workflow ID searches over the database tables for entities (processes) that have matching IDs. If there are dropped messages, the queried graph may have missing edges or missing vertices. The only guarantee for the retrieved graphs is that the components of the graph are linked through that workflow ID.

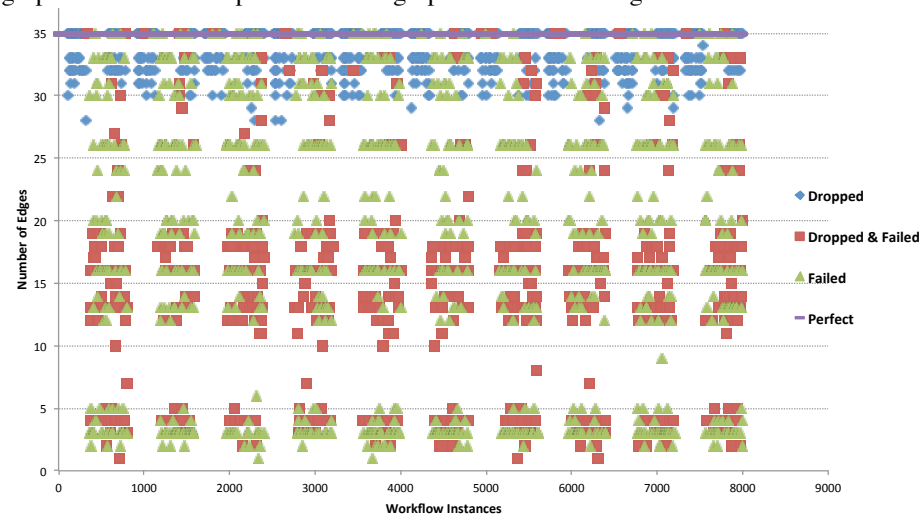

Fig. 6. Plot of edge counts for LEAD NAM workflow instances with different statuses.

In early results, we ran the algorithm against the 10GB provenance database and show the results in Figure 6 for the LEAD NAM workflow preliminary, observing the

number of edge counts for each workflow instance. The plot points are classified based on the statuses of each workflow. As one would expect, the perfect workflows have the complete 35 edges. We observe that workflows with dropped messages cluster towards the upper end of Figure 6. This implies that dropped messages for successful workflows are few. In comparison, workflows that involve failures typically result in more missing notifications, resulting in lesser number of edges in their provenance graphs. We also note that approximately 30% of imperfect workflows possess the full 35 edges. This is due to Karma not taking into account some of WORKEM's notifications, such as computation start and stop notifications and response status notifications. Though these workflows experience missing notifications when generated through WORKEM, the provenance graphs extracted from Karma, which construct graphs based on the objects and edges defined in the Open Provenance Model, appear to be perfect.

Automatic Provenance Repair. We are investigating the use of artificial intelligence methods to repair faults in provenance traces. We have developed a system, Phala [8] that uses case-based reasoning [10] from similar known workflows and additional methods to predict the missing steps in a partial workflow. Case-based reasoning systems reason from specific prior examples, solving new problems by retrieving records of prior problem-solving and adapting their solutions to fit new circumstances. Given an incomplete provenance trace, Phala retrieves prior traces involving similar steps, and predicts the missing steps by analogy to the provenance information in the prior traces. Phala's approach is strongly data-driven, relying on the database of previously-observed provenance rather than on knowledge-intensive analysis. As the pool of relevant prior provenance traces grows through provenance capture, so does the system's ability to suggest suitable repairs. Even incomplete stored provenance traces may be useful if they are, locally, more complete than the target trace. Note that the provenance database Phala uses need not be restricted to a single domain; its retrieval/similarity assessment criteria select relevant cases.

For large databases of cases, controlling retrieval/similarity assessment cost is a key issue. This is particularly important for structured cases such as workflows, in which similarity calculations must take into account structural similarity of the workflow graphs. To avoid the expense of full graph matching over the stored data, Phala uses a two-step retrieval process. The first phase uses coarse-grained criteria to retrieve a set of initial candidates from the full database, restricting structural considerations to small independent sub-structures over which cases are indexed; the second phase considers the complete structure of each case retrieved from the first phase and re-ranks cases accordingly. To improve accuracy and robustness to noise, Phala uses multiple reasoning techniques to generate predictions and reconciles divergent predictions through a confidence-weighted voting scheme [9].

Phala's approach has been tested for aiding users at incrementally extending a workflow during initial workflow construction (see [8] for results of an evaluation of accuracy and scalability). Provenance repair is a natural application for the system, but the size of provenance databases far exceeds that of datasets to which case-based reasoning has previously been applied. The 10GB provenance database provides a challenging testbed for future study and refinement of Phala's methods for handling large-scale provenance sources.

9 Conclusions and Future Work

In this paper, we present our methodology behind building a 10GB noisy provenance database, and the reasons why its existence is important. This sizable database consists of a varied distribution of realistic workflows. We provide details of our methodology for populating the database and provide evaluations of this workload database in terms of its distribution and performance. We plan to release this provenance database in the near future. It will be made available at: http://pti.iu.edu/d2i/provenance_gigabyte_database.

We are now using the provenance database to study provenance quality assessment. Our current graph analysis algorithm for this task makes simplifying assumptions about the existence of a workflow ID that ties together all notifications belonging to a provenance record. We plan to explore loosening this restriction. In addition, in order to determine how close a provenance record comes to a perfect record, one needs some sense of what is expected. This can be done by requiring a workflow template, which is realistic in some provenance capture settings but not others, or will require learning algorithms that can build a sense over time of a good provenance record. Finally, we are exploring using the graph structure to propagate node and edge quality metrics through the provenance graph.

Acknowledgements

We thank Karma team members Yiming Sun, Mehmet Aktas and Girish Subramanian. This work funded in part by National Science Foundation NSF-OCI-6721674 and by a grant from the Data to Insight Center of Indiana University. This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

1. S. Antonatos, K. Anagnostakis, and E. Markatos, Generating realistic workloads for network intrusion detection systems. In *ACM Workshop on Software and Performance*, Redwood Shores, CA, USA, 2004.
2. R. R. Bodnarchuk and R. B. Bunt, A synthetic workload model for a distributed systems file server. In *Proceedings of the SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp. 50–59, 1991.
3. Y. Cui and J. Widom, Lineage tracing for general data warehouse transformations, in *VLDB Journal*, vol. 12, 2003, pp. 41-58.
4. J. Freire, D. Koop, E. Santos, and C. T. Silva, 2008, Provenance for Computational Tasks: A Survey, *Computing in Science and Engineering*, vol.10, no.3, pp.11-21.
5. J. Frew, D. Metzger, and P. Slaughter. Automatic capture and reconstruction of computational provenance. *Concurrency and Computation: Practice and Experience*, 20(5): 485-496, 2008.
6. P. Groth and L. Moreau, Recording Process Documentation for Provenance, *IEEE Transactionson Parallel and Distributed Systems*, 20(9): 1246-1259, 2009.

7. J. Kim, E. Deelman, Y. Gil, G. Mehta, V. Ratnakar, Provenance Trails in the Wings/Pegasus System, *Concurrency and Computation: Practice and Experience*, 20(5): 587-597, 2008.
8. D. Leake and J. Kendall-Morwick, Towards Case-Based Support for e-Science Workflow Generation by Mining Provenance, *Advances in Case-Based Reasoning*: ECCBR 2008, Springer Verlag, Berlin, 2008, pp. 269-283.
9. D. Leake and J. Kendall-Morwick, Four Heads are Better than One: Combining Suggestions for Case Adaptation. *Case-Based Reasoning Research and Development: Eighth International Conference on Case-Based Reasoning,* ICCBR-09, LNAI 5650, Springer Verlag, Berlin, 2009, pp. 165-179.
10. R. Lopez de Mantaras, D. McSherry, D. Leake, B. Smyth, S. Craw, B. Faltings, M.L. Maher, M. Cox, K. Forbus, M. Keane, A. Aamodt, I. Watson, Retrieval, Revision, and Retention in CBR, *Knowledge Engineering Review*, 20(3), 215-240, 2006.
11. B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao, Scientific Workflow Management and the Kepler System, *Concurrency and Computation: Practice & Experience*, *Special Issue on Scientific Workflows*, 2005.
12. P. Mehra and B. Wah, Synthetic Workload Generation for Load-balancing Experiments, *IEEE Parallel and Distributed. Technology*, 3(3): 4–19, 1995.
13. L. Moreau, B. Plale, S. Miles, C. Goble, P. Missier, R. Barga, Y. Simmhan, J. Futrelle, R. McGrath, J. Myers, P. Paulson, S. Bowers, B. Ludaescher, N. Kwasnikowska, J. Van den Bussche, T. Ellkvist, J. Freire, and P. Groth, The Open Provenance Model. *Technical report, Electronics and Computer Science, University of Southampton*, 2008.
14. B. D. Noble, M. Satyanarayanan, G. T. Nguyen, and R. H. Katz, Trace-Based Mobile Network Emulation. In *Proceedings of SIG-COMM '97*, pp. 51–61, Cannes, France, September 1997.
15. L. Ramakrishnan and B. Plale, A Multi-Dimensional Classification Model for Workflow Characteristics, *Workflow Approaches to New Data-centric Science, with ACM SIGMOD 2010*, Indianapolis, IN.
16. L. Ramakrishnan, B. Plale, and D. Gannon, WORKEM: Representing and Emulating Distributed Scientific Workflow Execution State, *Proceedings of the 10th IEEE/ACM Int'l Symposium on Cluster, Cloud and Grid Computing*, Melbourne, Australia, 2010.
17. S. Shirasuna, A Dynamic Scientific Workflow System for the Web Services Architecture. PhD thesis, Indiana University, September 2007.
18. Y. Simmhan, B. Plale, and D. Gannon, Karma2: Provenance Management for Data Driven Workflows, *International Journal of Web Services Research*, IGI Publishing, vol. 5, no.2, 2008.
19. Y. Simmhan, B. Plale, D. Gannon, Towards a Quality Model for Effective Data Selection in Collaboratories, *IEEE Workshop on Workflow and Data Flow for Scientific Applications*, *held in conjunction with ICDE*, Atlanta, GA, 2006.
20. Y. Simmhan, Beth Plale, Dennis Gannon. "A survey of data provenance in e-science", SIGMOD Record 34(3): 31-36, 2005.
21. K. Sreenivasan, A.J. Kleinman, On the construction of a representative synthetic workload, *Communications of the ACM,* 1974, pp.127-133.
22. J. Widom, "Trio: A System for Integrated Management of Data, Accuracy, and Lineage," in *CIDR*, Pacific Grove, California, January 2005.