How much domain data should be in provenance databases?

Daniel de Oliveira
Instituto de Computação
Universidade Federal Fluminense
Niterói, Brazil
danielcmo@ic.uff.br

Vítor Silva
COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
silva@cos.ufrj.br

Marta Mattoso
COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
marta@cos.ufrj.br

ABSTRACT

Provenance databases are an important asset in data analytics of large-scale scientific data. The data derivation path allows for identifying parameters, files and domain data values of interest. In scientific workflows, provenance data is automatically captured by workflow systems. However, the power of provenance data analyses depends on the expressiveness of domain-specific data along the provenance traces. While much has been done through the W3C PROV initiative and its PROV-DM to represent generic provenance data, representing domain-specific data in provenance traces has received little attention, yet it accounts for a large number of provenance analytical queries. Such queries are based on selections on data values from input/output artifacts along workflow activities. There are several problems in modeling and capturing values from domain-specific attributes, some of them are related to managing provenance granularity, others to addressing data values hidden inside files and representing the semantics of domain data. In this work, we discuss these open issues and propose some alternatives to domain-specific provenance data capture, representation, storage and queries. Addressing these issues may be decisive in using provenance to drive scientific data analyses at large-scale.

1. INTRODUCTION

Scientific Workflow Management Systems (SWfMS) have been helping on several scientific data analysis. SWfMS with parallel processing such as Pegasus [1], Swift/T [2] and SciCumulus [3], automatically partition data input and manage the dataflow generation in parallel processing data at large scale, in clusters or clouds. In addition to providing a high scalability, parallel SWfMS systems can improve scientific data analyses through provenance database query support. Provenance data from scientific workflows represent the data derivation path, which allows for identifying parameters, files and domain data values of interest [4].

Domain-specific data analysis account for most of provenance queries found in the Provenance Challenge series and in reports with real scientific workflows [5,6]. The data analytical power of provenance depends on the expressiveness of domain data in provenance traces. For example, in bioinformatics (e.g. SciPhy, a workflow for Phylogenetic analysis [7]), workflows may involve genome sequence similarity analysis programs that register the resulting similarity value inside an output file. In this case, the required domain data value is outside the scope of the workflow provenance traces. As observed by Alper *et al.* [5], many provenance queries are based on data values that are "assumed" to exist. Even when they do exist, in the provenance trace, it may be difficult to reference domain-specific data in a query. There are several open issues in querying domain data from provenance databases. Basically these challenges are related to domain data values that must appear explicitly in the provenance trace; and values that have to be addressable by a provenance query language, *i.e.*, there must be an access path to be used in searching for the required domain value. We discuss these two issues as follows.

The first issue is related to defining how much domain data should be represented in provenance traces. This is highly related to provenance granularity, which has been discussed in several papers, such as: [8], [9], [10]. Provenance in coarse grain is represented generically as input artifacts transformed by activities that generate output artifacts. Individual domain-data values are not represented in coarse-grain, unless they are explicitly defined as an activity output. Many relevant domain-specific values are "hidden" inside input/output raw data files. In this case, provenance database queries may just obtain a specific file identifier. Then, this file has to be further analyzed, based on its contents. In this scenario, provenance data acts as pointers to raw data files generated from the workflow execution. To analyze these raw data file contents, the user has to write a program or use specific data analyses tools [11,12], which are disconnected from the data derivation path and other provenance data.

The second issue refers to the difficulty in representing semantics of domain data, so they can be accessed. This occurs more frequently in fine-grain provenance capture. In most fine-grain representations, data is typically captured at the operating system level, which generates a data deluge and these provenance traces are very difficult to be queried. However, independent of provenance granularity, it is not simple to address domain-specific data from provenance traces in query specification. To improve domain data provenance query capabilities, LabelFlow [5] suggests adding annotations/labels to data artifacts from provenance traces. However, these labels are added only after the provenance trace is generated. In a large-scale parallel environment, runtime provenance queries are often required to monitor, debug and steer the workflow execution [6].

The approach we have been taking lies in adopting an intermediate provenance representation level between coarse and fine grain provenance capture [13]. In addition to workflow coarse grain data artifacts, special domain data are selected by the user to be represented in provenance traces. Domain data capture activities are included in the workflow definition so that these

TaPP'15, July 8–9, 2015, Edinburgh, Lothian, Scotland.

domain data values get to be related to other provenance data. Just like debugging and execution profiling tools, the user sets which domain data values should be registered for further inspection. This setting may be driven by the PrIME methodology [14], which presents a systematic approach with phases that help guiding the user in defining the provenance queries that would then define relevant domain data values to be captured.

In [15] we presented the PROV-Wf model, which is a W3C PROV-DM [16] specialization. PROV-Wf models generic workflow provenance, domain-specific data and execution data, all related in the same model. ProvONE [17] is also a PROV extension to represent scientific workflow classes, but representing and addressing domain data seems to remain an open issue in ProvONE. We believe that there should be more efforts towards these provenance representations, particularly with respect to domain data extraction, representation and queries to improve scientific analyses and provenance data interoperability. However, the answer to how much domain data should be in provenance databases remains an open issue. It depends on the analytical target of the user and the overhead in collecting the corresponding data. In this work, we present our efforts in this direction by showing extensions to our PROV-Wf model to improve domain-data management; show the power of data analytics in provenance queries; and discuss current challenges also in light of the state of the art proposals.

This paper is organized as follows. Section 2 discusses related work. Section 3 presents PROV-Wf extensions and analyzes its concepts with respect to the ProvONE data model. Section 4 shows queries from a case study based on the instantiation of PROV-Wf in a real workflow. Finally Section 5 concludes this paper and points out future work.

2. RELATED WORK

In this section, we analyze related work on the provenance management issues involved in domain data capture, representation and queries. Provenance capture in different levels of granularity has been addressed by many related work [8], [9], [10]. However, the role of domain data has not been very much discussed in workflow provenance representation. We are particularly concerned with relevant domain data from raw data file contents, "hidden" from the provenance management system. Yet, to improve scientific data analysis, these domain data values must appear explicitly in the provenance trace; and these values have to be addressable by a provenance query language. The following papers address issues of domain data provenance capture. Due to similarities in scientific workflow modeling between PROV-Wf and the ProvONE [17] data model proposal for scientific workflow provenance, we left it to be discussed in more detail on Section 3. Nevertheless, we did not find any related work discussing domain-specific data storage and querying in provenance for large-scale systems.

In [9] the authors show the impact of the provenance capture method on the resulting provenance trace. They show that this difference in capturing methods makes it difficult to find equivalent tasks or data. In our work, we share the same concerns towards interoperability of provenance traces and include an additional effort to capture and represent domain-specific data in provenance traces, so that domain-data provenance queries become more uniform.

Perm [18] is a provenance system which also considers the importance of having domain-specific data and provenance data all in the same model and database. For example, Glavic *et al.* [19] present some SQL queries (for types *Why, Where* and *How*) using Perm for analyzing domain-specific data associated to provenance data. However, Perm is focused on database SQL query provenance, where relational algebraic transformations generate data that is naturally represented in databases. Therefore, combining domain data with provenance data in relational databases seems more natural. In the case of scientific workflows, domain data is typically in raw data files, which are not suited to be loaded in structured databases. Still there should be an effort in selecting some of this raw data to be captured, structured and related to their corresponding provenance data, all represented in the same model and stored in a single provenance database. The same advantages of having complex queries with optimizations from relational database systems mentioned by Glavic and Alonso in [18] apply here.

SPADE [10] also focus on having user input to select relevant domain data to be captured to obtain a provenance database that lies between coarse and fine grain provenance. SPADE allows for defining restrictions in this domain capture to decrease overheads related to storage and querying. In relation to our paper, SPADE does not consider data gathering from raw data file contents and does not mention provenance query processing at runtime. In the scientific workflow scenario of large-scale with parallel processing, many raw data files are generated, which need to be queried during the workflow evolution.

3. PROV-WF DATA MODEL

It is well-known that W3C PROV recommendation [20] allows for representing entities, people and processes involved in the generation of a piece of data. PROV aims at enabling the representation of provenance information in heterogeneous environments. PROV-DM is able to model all types of provenance, thus not being specific for representing provenance of scientific workflows. In this way, we presented PROV-Wf [15] as an extension to PROV-DM to model workflow execution data and domain-specific data to improve provenance analysis.

PROV-Wf is the basis of existing SWfMS, such as Chiron [13] and SciCumulus [3] and has been used in several real workflows. In this section, we give a brief overview of PROV-Wf and focus on its extensions for representing and querying "hidden" domain-specific data. Recently, another PROV-DM extension has been proposed such as ProvONE [17] and it is aimed at being a referenced model to scientific workflow provenance systems. Therefore, we also include an effort towards identifying common aspects between PROV-Wf and ProvONE and discuss open issues with respect to domain data.

PROV-Wf represents prospective and retrospective provenance data for scientific workflows. The extended version of PROV-Wf is composed of four main parts (Figure 1): the structure of the scientific workflow (white classes in the UML class diagram); the execution of scientific workflow (dark gray classes); domain data (dark gray classes); and the environment configuration (light gray classes).

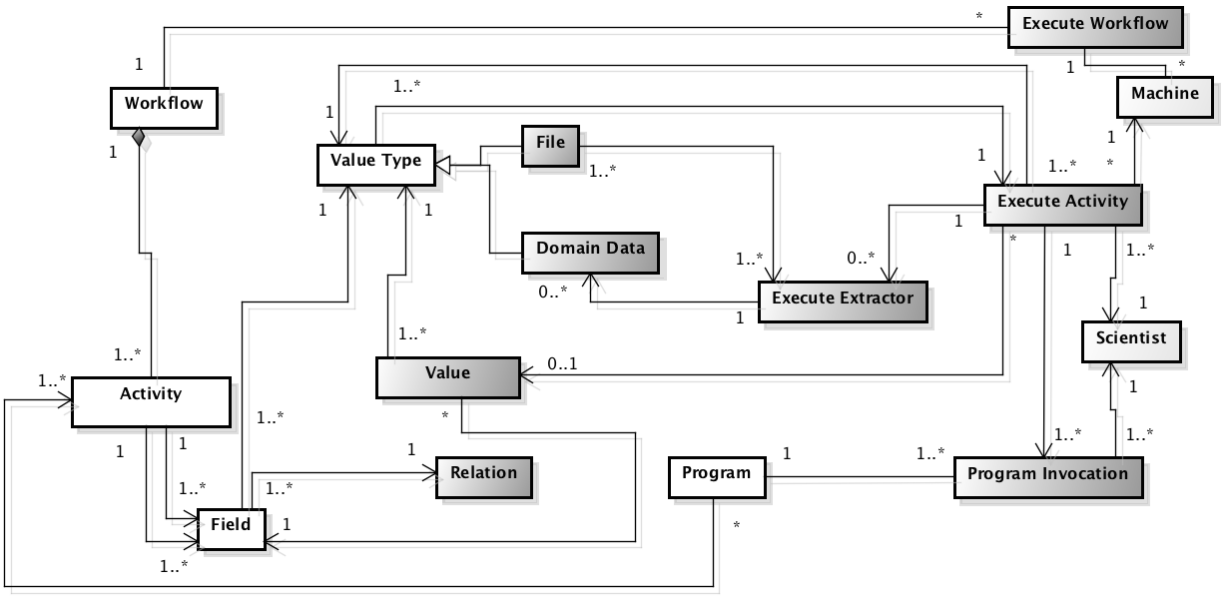

Figure 1. Extended version of PROV-Wf data model from [15]

In this extension of PROV-Wf, a scientific workflow (class *Workflow*) is composed by a set of activities (*i.e.* class *Activity*). Each activity is associated with a program that implements it (class *Program*). The execution of an activity (class Execute Activity) invokes a specific program within a workflow (class *Program Invocation*) consuming a set of parameters (class Field) that can be seen as a set of values to be consumed (class *Value*). To express all data that is consumed and produced by an activity execution (class *Execute Activity*), the class *Relation* represents the metadata that describes all explicit data (class *Field*), *i.e.* all parameters consumed and produced by a specific activity. The class *Value* expresses the value for a specific parameter associated to a specific activity execution. Furthermore, each value presents a specific data type or structure (class *Value Type*) that can be a file (class *File*) or domain-specific data (class *Domain Data*). These classes act as generic classes to be specialized by domain data classes, having names of relations and attributes that are defined by the user. PROV-Wf also considers that a program can be triggered by an activity execution in order to collect domain-specific data from produced raw data files. This program invocation is described by an extractor execution (class *Execute Extractor*), which consumes a raw data file and produces domain-specific data. These extracted domain data values are loaded into domain specific relations. According to the environment configuration, each activity execution gathers information about computational resources (class *Machine*) and the users that invoked a specific activity or program (class *Scientist*). With the class *Machine*, we consider that information about computational resources need to be gathered during the workflow execution, once the user may need to query about the machine configuration to compare different workflow executions.

In the same way, ProvONE proposes an extension of PROV data model also representing prospective and retrospective scientific workflow provenance. ProvONE is composed by a set of classes that represent both the structure of the workflow and the consumed and produced data. In ProvONE, the various tasks that are part of a specific workflow are represented by the class *Program*. These programs are single ones or can be composed of other programs (*i.e.* composite). A specific composite program is defined as a *Workflow*. Each *Program* has one or more *Ports* (input or output) to represent data that is consumed and produced. These ports are connected through PROV *Locations*. The program executions of a specific workflow are represented in ProvONE using the class *Execution*. The instances of the class *Execution* represent the execution of a program (that can be a workflow), and are associated with a *User* that performs the execution. Representing execution data, such as begin time and end time of a program execution is found in both PROV-Wf and ProvONE. These are important information for workflow data analytics related with provenance classes, as shown in the domain data queries of the next section.

In ProvONE, for each execution a series of input *Data* items are read from the input *Ports* and are used to generate a series of output *Data* items sent through the output *Ports*. The types of outputs are *Data*, *Visualization*, or *Document* items. The class *Data* represents data of various types. *Visualization* is a differentiated class intended to represent images produced as output from workflows. The *Document* class is a generic representation of a published or unpublished article of a given execution of the workflow. A *Collection* class may in turn represent a set of the previously mentioned output types.

While ProvONE represents file tracing between program executions, the extended version of PROV-Wf adds selected domain data extracted from these files, keeping the derivation relationship between files and elements from the files. Another difference between PROV-Wf and ProvONE is related to the *Activity* class. ProvONE presents a workflow as a composition of programs, while PROV-Wf specifies a workflow by a set of activities that are in turn associated with programs. Each activity presents one or more program in its specification, whose definition is represented by the relationship *wasAssociatedWith* between *Activity* and *Program* classes. Thus, we believe that activity and program are different concepts and should to be modeled in different classes. This use of the class *Activity* also keeps PROV-Wf compatible with PROV-DM.

4. PUTTING A UNIFIED PROVENANCE MODEL IN PRACTICE

This section aims at presenting the potential of the extended version of PROV-Wf, which integrates data about workflow structure, workflow execution, and domain data. We present an instantiation of PROV-Wf classes in a provenance relational database of a SWfMS, for SciPhy [7], a bioinformatics workflow for phylogenetic analyses that aims at producing phylogenetic trees to represent existing evolutionary relationships. Then, we present some provenance queries involving data from workflow structure, execution, and domain.

In the instantiated provenance relational database schema[1], each table is associated with a class of PROV-Wf model. The mappings between PROV-Wf classes and the provenance relational schema are presented in Table 1. Each row in Table 1 has a background color, and each color is associated with a specific type of table in the relation schema: tables associated to workflow representation or environment configuration (white); tables generated for representing consumed and produced data by each activity (light gray); and tables for extracted domain-specific data (dark gray).

A workflow (table *cworkflow*) is composed of one or more activities, which present information about program invocation, domain data extraction and some additional operations for joining results (respectively, tables *cactivity, cextractor, coperand* and *cjoin*). Each activity is represented by the consumption and production of some relations (table *crelation*), which present some data dependencies (table *cmapping*) with fields (table *cfield*) from other relations. The value type of each field is also specified in table *cfield*.

Table 1. PROV-Wf classes in a provenance database

PROV-Wf class	Table
Workflow	cworkflow
Activity	cactivity, coperand, cextractor, cjoin
Relation	crelation, cmapping
Field	cfield
Program	eactivity
Execute workflow	eworkflow
Execute activity	eactivity
Execute extractor	eactivation
Machine	emachine
Scientist	-
Program invocation	eactivation
File	efile
Value	idataselection, odataselection, omafft, oreadseq, omodelgenerator, oraxml1, oraxml2, omergeraxml, oraxml3
Value type	cfield
Domain Data	dlmafft, dlreadseq, dlmodelgenerator, dlraxml1, dlraxml2, dlmergeraxml, dlraxml3

Considering workflow execution (tables *eworkflow*), provenance data about activity executions are stored in table *eactivity*, such as program specifications. Meanwhile, data about program invocation and extractor execution are stored in table *eactivation*. For environment configuration, our relational database stores properties about computational resources in table *emachine*. Currently, class *Scientist* is not represented in the provenance relational database yet.

[1] http://cos.ufrj.br/~silva/relational-database-schema-SCC.png

The highlighted lines in Table 1 show classes related to domain-specific data and the tables that instantiate theses classes have names given by the user while modeling the workflow. In this case, it shows tables and attributes corresponding to the SciPhy workflow, presented in Figure 2. Data dependencies are represented by arrows between activities in Figure 2 and implement *Influence* component in PROV-DM. These tables have the same name of the workflow specification (*e.g.* table *idataselection*). The produced domain-specific files are stored in table *efile*, while the extracted domain data is stored in tables for extracted values, such as *dlmafft*.

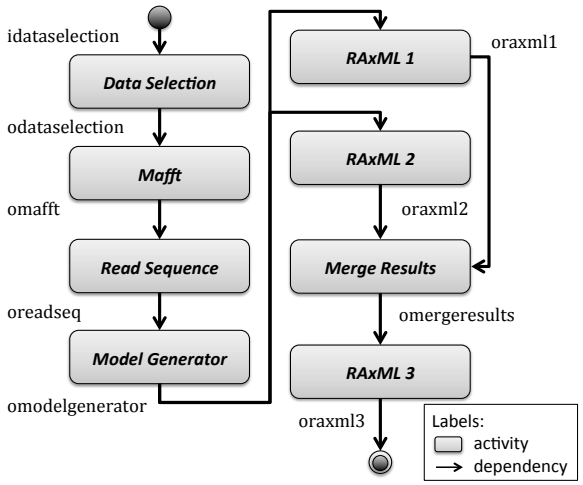

Figure 2. SciPhy workflow

Based on this provenance relational schema, we discuss a few queries that take advantage of the integrated provenance database, particularly with domain data. In the first query, users may need to identify where are the produced files of each task that runs the *ModelGenerator* program and present the average execution time higher than 3 standard deviations. This will define the region of the parameter space to be investigated. This query is presented in Figure 3. This query can be classified in the types WHERE and WHEN from Buneman et al. [21] classification of provenance queries. These types of query can be performed in most of the provenance databases, independent if they follow the PROV-Wf, ProvONE, or others that combine execution data with generic provenance data.

In a parallel execution, as it is the case of the SciPhy workflow, input data is partitioned and several program instances are executed as parallel tasks. Users often need more than file names to evaluate what is "going on" with some parallel tasks. They commonly have to analyze the content of files, *i.e.* domain-specific data. If each task with anomalous execution time produces many files, it may be hard for users to browse all files to parse domain-specific data within them. This browsing task may be hard, tedious and error-prone. For example, in a real execution of SciPhy, for 200 input files, one tuple is inserted in *eworkflow* table, eight corresponding tuples in *eactivity*, 1,406 in *eactivation*, 9 in *crelation*, 19,419 in *efile,* and 1,290 in domain tables. This gives an idea of the number of files that are generated. To select the corresponding file name that presents a bootstrap value greater than a threshold, it would require creating a specific program for opening and parsing the required files for this analysis. This would be just part of the single raw data file analysis. The user might also want to trace the corresponding files (and some of their attributes) generated by the next activities that received these

"selected" bootstrap values as input. This analysis, when complemented with execution time behavior is important for the trials involving the workflow configuration exploration or debugging. Before achieving a final configuration, this workflow execution trial might show that some of the parameters on this workflow configuration were not adequate for this specific input dataset and the execution must be interrupted for another trial. Some more specific analytical queries are shown as follows.

```
SELECT t.taskid, f.fdir, f.fname
FROM eactivity a, eactivation t, efile f
WHERE a.actid = t.actid
AND t.taskid = f.taskid
AND a.tag = 'modelgenerator'
AND round(cast (extract(epoch from (t.endtime - t.starttime)) AS
numeric),2) >
    (
    SELECT (avg(round(cast (extract(epoch from (t.endtime -
    t.starttime)) AS numeric),2)) + 3*stddev(round(cast (extract(epoch
    from (t.endtime - t.starttime)) AS numeric),2))) as threshold
    FROM eactivity a, eactivation t
    WHERE a.actid = t.actid
    AND a.tag = 'modelgenerator'
    );
```

Figure 3. Query from categories where and when

Figure 4 presents a typical analysis that the user does in SciPhy. One way of knowing if a specific task is presenting anomalous behavior is having the users analyzing the number of alignments produced in the alignment activity. This number is used as input for *ModelGenerator* (attribute *num_aligns*), and allows obtaining the length of the biggest sequence in the dataset (attribute *length*) and the evolutionary model used to infer the evolutionary relationships among sequences (attribute *model1*). Note that attributes *length* and *num_aligns* were extracted from data files of the previous activities of the workflow and then propagated to the input relation of *ModelGenerator* activity. This allows for analyzing domain-specific data elements using its data derivation path.

```
SELECT t.taskid, dlmg.model1, dlrs.num_aligns,
            dlrs.length
FROM eactivity a, eactivation t, sciphy.omodelgenerator omg,
            sciphy.dlmodelgenerator dlmg, sciphy.dlreadseq dlrs
WHERE a.actid = t.actid
AND a.tag = 'modelgenerator'
AND omg.previoustaskid = t.taskid
AND omg.dlmodelgeneratorid = dlmg.rid
AND omg.dlreadseqid = dlrs.rid
AND round(cast (extract(epoch from (t.endtime - t.starttime)) AS
numeric),2) >
    (
    SELECT (avg(round(cast (extract(epoch from (t.endtime -
    t.starttime)) AS numeric),2)) + 3*stddev(round(cast (extract(epoch
    from (t.endtime - t.starttime)) AS numeric),2))) as threshold
    FROM eactivity a, eactivation t
    WHERE a.actid = t.actid
    AND a.tag = 'modelgenerator'
    );
```

Figure 4. Query from categories when and why

In addition, users may need to investigate a more restricted parameter space. Thus, they normally need to reduce their parameter space by some domain-specific data. For example, users may analyze only the best scores (attribute *bestscore*) that present a bootstrap greater or equal to 25 and an alignment length greater than 60 in SciPhy, as presented in Figure 5.

5. CONCLUSION

Provenance models are domain agnostic. However, most analytical queries in scientific workflow data involve domain-specific concepts. In this work we discuss some provenance representation alternatives towards defining a provenance data model, which represents domain-specific data explicitly in the provenance trace and addresses domain-specific data by a provenance query language. To avoid the extremes of coarse versus fine grain provenance capture our approach lets the user indicate which and how much domain data should be captured, while the system automatically captures and represents them with the generic provenance data. However, ideally the system should suggest which data and evaluate how much data capture would penalize the performance and storage of workflow execution in light of the benefits in data analytics.

We presented an extended version of our PROV-Wf model with the purpose of supporting the issues of domain data provenance. We propose some new classes including the domain-specific data (*Domain Data* class), where the user chooses the attribute and relation names, the different domain-data value types (*Value Type* class), and the extraction of domain-specific data from raw data files (*Execute Extractor* class). By addressing these classes on the provenance database, the user, in our bioinformatics example, may obtain domain data values for the number of alignments resulting from activity "model generator" with a specific filter. Otherwise, the user would have to obtain the corresponding output files and open them to extract the number of alignments. However, we believe that there should be a joint effort towards domain aware provenance data models so that provenance queries improve the analytical power at the same time it may interoperate with different provenance databases.

```
SELECT t.taskid, dlr1.besttree
FROM eactivity a, eactivation t, sciphy.oraxml3 or3,
        sciphy.dlraxml3 dlr3, sciphy.dlreadseq dlrs,sciphy.dlraxml1 dlr1
WHERE a.actid = t.actid
AND a.tag = 'raxml3'
AND or3.taskid = t.taskid
AND or3.dlraxml3id = dlr3.rid
AND or3.dlraxml1id = dlr1.rid
AND dlr3.minbootstrap >= 25
AND dlrs.length > 60
AND round(cast (extract(epoch from (t.endtime - t.starttime)) AS
numeric),2) >
    (SELECT
    (avg(round(cast (extract(epoch from (t.endtime - t.starttime)) AS
    numeric),2)) +
    3*stddev(round(cast (extract
    (epoch from (t.endtime - t.starttime)) AS numeric),2))) as
    threshold
    FROM eactivity a, eactivation t
    WHERE a.actid = t.actid
    AND a.tag = 'modelgenerator')
```

Figure 5. Query from categories where, when and why

6. ACKNOWLEDGMENTS

We thank Kary Ocaña for her help on configuring and executing SciPhy programs. We also acknowledge the Brazilian funding agencies CAPES, CNPq and FAPERJ for partially sponsoring this work and NACAD Center for the parallel computing support.

7. REFERENCES

[1] J. Kim, E. Deelman, Y. Gil, G. Mehta, and V. Ratnakar. Provenance trails in the Wings-Pegasus system. Concurrency and Computation: Practice & Experience, 20:587–597, 2008.

[2] J.M. Wozniak, T.G. Armstrong, M. Wilde, D.S. Katz, E. Lusk, and I.T. Foster. Swift/T: Large-Scale Application Composition via Distributed-Memory Dataflow Processing. Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 95–102, 2013.

[3] D. Oliveira, E. Ogasawara, F. Baião, and M. Mattoso. SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. Proceedings of the 3rd International Conference on Cloud Computing, 378–385, 2010.

[4] S.B. Davidson and J. Freire. Provenance and Scientific Workflows: Challenges and Opportunities. ACM SIGMOD International Conference on Management of Data, 1345–1350, 2008.

[5] P. Alper, K. Belhajjame, C.A. Goble, and P. Karagoz. LabelFlow: Exploiting Workflow Provenance to Surface Scientific Data Provenance. Provenance and Annotation of Data and Processes, B. Ludäscher and B. Plale, eds., Springer International Publishing, 84–96, 2015.

[6] M. Mattoso, J. Dias, K.A.C.S. Ocaña, E. Ogasawara, F. Costa, F. Horta, V. Silva, and D. de Oliveira. Dynamic steering of HPC scientific workflows: A survey. Future Generation Computer Systems, 46:100–113, 2015.

[7] K.A.C.S. Ocaña, D. de Oliveira, E. Ogasawara, A.M.R. Dávila, A.A.B. Lima, and M. Mattoso. SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes. Advances in Bioinformatics and Computational Biology, O.N. de Souza, G.P. Telles, and M. Palakal, eds., Springer Berlin Heidelberg, 66–70, 2011.

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

[9] B. Coe, R.C. Doty, M.D. Allen, and A. Chapman. Provenance Datasets Highlighting Capture Disparities. Proceedings of the 6th Workshop on the Theory and Practice of Provenance (TaPP), 2014.

[10] D. Tariq, M. Ali, and A. Gehani. Towards Automated Collection of Application-Level Data Provenance. Proceedings of the 4th USENIX Workshop on the Theory and Practice of Provenance (TaPP), 2012.

[11] M. Karpathiotakis, M. Branco, I. Alagiannis, and A. Ailamaki. Adaptive Query Processing on RAW Data. Proceedings of the VLDB Endowment, 7(12):1119–1130, 2014.

[12] J. Chou, K. Wu, and Prabhat. FastQuery: A Parallel Indexing System for Scientific Data. Proceedings of the 2011 IEEE International Conference on Cluster Computing (CLUSTER), 455–464, 2011.

[13] E. Ogasawara, J. Dias, D. Oliveira, F. Porto, P. Valduriez, and M. Mattoso. An Algebraic Approach for Data-Centric Scientific Workflows. Proceedings of the 37th International Conference on Very Large Data Bases (PVLDB), 4(12):1328–1339, 2011.

[14] S. Munroe, S. Miles, L. Moreau, and J. Vázquez-Salceda. PrIMe: a software engineering methodology for developing provenance-aware applications. Proceedings of the 6th international workshop on Software engineering and middleware, 39–46, 2006.

[15] F. Costa, V. Silva, D. de Oliveira, K. Ocaña, E. Ogasawara, J. Dias, and M. Mattoso. Capturing and Querying Workflow Runtime Provenance with PROV: A Practical Approach. Proceedings of the Joint EDBT/ICDT 2013 Workshops, 282–289, 2013.

[16] L. Moreau and P. Missier. PROV-DM: The PROV Data Model. Available at: http://www.w3.org/TR/2013/REC-prov-dm-20130430/ Accessed: 17 Feb 2014., 2013.

[17] ProvONE. Available at: http://jenkins-1.dataone.org/jenkins

[18] B. Arab, D. Gawlick, V. Radhakrishnan, H. Guo, and B. Glavic. A Generic Provenance Middleware for Queries, Updates, and Transactions. Proceedings of the 6th Workshop on the Theory and Practice of Provenance (TaPP), 2014.

[19] B. Glavic, R.J. Miller, and G. Alonso. Using SQL for Efficient Generation and Querying of Provenance Information. In Search of Elegance in the Theory and Practice of Computation, V. Tannen, L. Wong, L. Libkin, W. Fan, W.-C. Tan, and M. Fourman, eds., Springer Berlin Heidelberg, 291–320, 2013.

[20] P. Groth and L. Moreau. Provenance: an introduction to PROV. Morgan & Claypool Publishers, 2013.

[21] P. Buneman, S. Khanna, and W. Tan. Why and Where: A Characterization of Data Provenance. International Conference on Database Theory:316–330, 2001.