Applying the Virtual Data Provenance Model

Yong Zhao[1], Michael Wilde[2], and Ian Foster[2]

[1] University of Chicago
yongzh@cs.uchicago.edu
[2] University of Chicago and Argonne National Laboratory

Abstract. In many domains of science, engineering, and commerce, data analysis systems are employed to derive new data (and ultimately, one hopes, knowledge) from datasets describing experimental results or simulated phenomena. To support such analyses, we have developed a "virtual data system" that allows users first to define, then to invoke, and finally explore the provenance of procedures (and workflows comprising multiple procedure calls) that perform such data derivations. The underlying execution model is "functional" in the sense that procedures read (but do not modify) their input and produce output via deterministic computations. This property makes it straightforward for the virtual data system to record not only the recipe for producing any given data object but also sufficient information about the environment in which the recipe has been executed, all with sufficient fidelity that the steps used to create a data object can be re-executed to reproduce the data object at a later time or a different location. The virtual data system maintains this information in an integrated schema alongside semantic annotations, and thus enables a powerful query capability in which the rich semantic information implied by knowledge of the structure of data derivation procedures can be exploited to provide an information environment that fuses recipe, history, and application-specific semantics. We provide here an overview of this integration, the queries and transformations that it enables, and examples of how these capabilities can serve scientific processes.

1 Introduction

We present a general model for representing and querying provenance information within the context of a Virtual Data System (VDS) that captures, and enables discovery of, the relationships among data, procedures and computations. We focus, in particular, on the VDS query model, and examine how knowledge of the provenance of virtual data objects and their relationships can be used to enhance program development, data analysis, and other tasks.

In what we call the *virtual data model*, we associate with each data object the functional procedure that was used, or can be used, to produce or reproduce it. Such associations are defined with sufficient fidelity that the steps used to create a data object can be re-executed to reproduce the data object (within obvious limitations) at a later time or a different location. We refer to the information that we record to achieve this reproducibility the *provenance* of a data object. (Throughout this article,

L. Moreau and I. Foster (Eds.): IPAW 2006, LNCS 4145, pp. 148 – 161, 2006.

we use the term "procedure" to denote executable application programs, but the paradigm applies equally well to a service-oriented model in which "procedures" correspond to invocations of remote operations.)

We view provenance in this context as comprising two parts: all the aspects of the procedure or workflow used to create a data object (*prospective* provenance, or "recipe") as well as information about the runtime environment in which a procedure was executed and the resources used in its invocation (*retrospective* provenance).

While only the prospective information is needed to produce or reproduce a data object, we argue that the *complete* provenance record—prospective and retrospective—provides a more complete understanding of the data. For instance, retrospective provenance can help investigate a data derivation process, as it keeps information regarding the environment in which the process was performed. This level of understanding is of great value in scientific data preparation and analysis, allowing the user to (for example) reason about the validity of data and conclusions drawn from it; determine and assess the methods that were used to process the data; and transform or compose existing methods to handle new problems.

The Virtual Data System that we have developed to implement this model [ZW+05] maintains a precise record of procedures, inputs (both data and parameter settings) to procedures, the environment in which procedures were invoked, and relevant data about how a procedure behaved (e.g., duration). Armed with this information, we can track, for any data object created within the system, a derivation history that extends back to raw input data, and thus obtain accurate and complete information about how analysis conclusions (and all intermediate results) were derived. We can understand data dependencies, and reason about the consequences for an analytical finding of changing some processing step, parameter, or input dataset. We can audit how results were derived, and create new recipes for conducting new investigations that build on previous findings and approaches.

An important component of VDS is the Virtual Data Language (VDL) [FV+02], a functional scripting language that we use to describe relations among data, procedures, and computations that invoke procedures. A data analysis workflow expressed in VDL makes the relationships among these different elements explicit.

VDS also incorporates sophisticated mechanisms for executing both individual procedures and more complex workflows in distributed environments. These mechanisms include tools for integrating data in diverse physical representations [ZD+05], workflow transformation tools and planners, such as the Pegasus system [DS+05, SKD06], the DAGman workflow execution system [FT+02], and Globus mechanisms for secure and reliable remote execution and distributed data management [F05]. This aspect of the system is less relevant to our goals here (except in that the transformations performed, and specific execution sites chosen, may be an important part of the provenance record), and so we do not discuss it further in this article.

VDS is distinguished from other approaches to provenance recording by its focus on a particular computational model, namely the functional model defined by VDL. While this focus restricts the set of computations that can be represented, we do not find this restriction to be onerous in practice, and the benefits in terms of the depth of provenance information that can be captured efficiently and the variety of queries that can be posed against that data are significant.

We focus in this article on illustrating the virtual data approach to integrating prospective and retrospective provenance with semantic annotations; describing the powerful queries that can be performed on such an integrated base; and introducing the implementation techniques that can provide these benefits in a large-scale scientific computing environment. The basic mechanisms for these techniques have been implemented in our Virtual Data System for some time [FV+02]; this paper describes schema extensions whose implementation is in progress.

2 Virtual Data Schema for Provenance Recording

We model as a logical virtual data schema the various relationships that exist among *datasets*, *procedures*, *calls* to procedures (which operate on datasets), and the zero or more physical *invocations* of a specific call. These relations are described by the entity-relationship (ER) diagram of Figure 1. In this diagram, primary keys are underlined; foreign keys are implied by graph edges.

A computational *procedure* represents an application or a service that can be executed or invoked. A procedure definition describes the procedure's signature: its name and formal arguments. A procedure may be defined in a specific namespace and may have different versions. Procedures are therefore identified by the 3-tuple (*namespace, name, version*). A formal argument (*FormalArg*) has a name, a type, and a direction attribute that indicates whether it is an input or output argument.

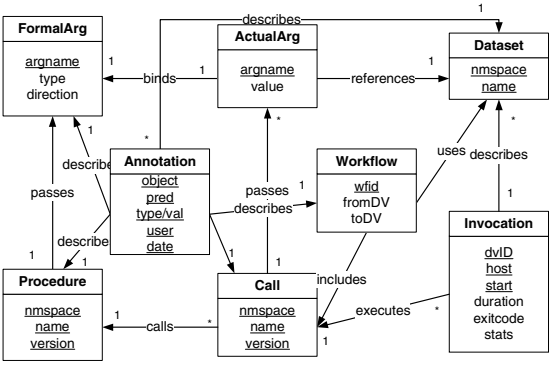

Fig. 1. Schema for provenance and annotation

A *call* specifies a procedure call, supplying actual arguments (*ActualArgs*) that bind values and datasets to formal arguments. Like procedures, calls are identified by (*namespace, name, version*) tuples, but for calls these unique tuples can be generated automatically. A call statement is considered prospective as it only declares a way to invoke a procedure, and specific data products can be generated by making the described procedure call. But by itself, a call is not executed until it is included in a *workflow* and scheduled to run. We use the term *derivation* (DV) interchangeably for a call.

We also model *workflows*, i.e., sets of calls that operate on the same datasets. A workflow is represented by one or more entries in the Workflow table, each entry for an edge of the workflow graph, which contains the source call (*fromDV*) that produces a certain data product, and the target call (*toDV*) that consumes that data product. A workflow itself, like a call, is prospective, and can be enacted multiple times. Each enactment of both a call and a workflow is recorded by an invocation record.

This integration of workflows enables queries to consider the provenance history of data objects, and the relationships among procedures based on the patterns in which they are actually used in defined workflows.

Metadata associates annotations with datasets (via their names), procedures and arguments, calls, and workflows. Annotations take the form of a named predicate and a typed value (a string, integer, float, boolean, or date). This simple type model for annotation values is readily extended to use, for example, the flexible data typing model defined by XML Schema. Such metadata annotations are similar to RDF triples: The *subject* is one of the 5 entities in the virtual data provenance model that can be annotated (datasets, procedures, etc.); the *predicate* is the "name" of the annotation (i.e., the assertion being made) and the value is the *object*. We plan to extend annotations with user and date information, so that we can maintain the provenance of metadata itself.

3 General Model for Provenance and Annotation Query

Having defined our virtual data model in relational terms, we can use standard SQL to query entities in the data model. For example, we can ask questions such as "select procedure calls whose argument *modelType* has value *nonlinear*," "select invocations that ran at location *Argonne*," and the join query "select procedure calls that ran at location *Argonne* and whose argument *modelType* has value *nonlinear*."

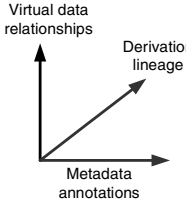

Fig. 2. Query dimensions

We find it useful to think of the virtual data query model as having three major dimensions (see Figure 2): (1) prospective and retrospective provenance data, as provided by records of procedure definition, procedure arguments, and runtime invocation recording; (2) metadata annotations that enrich this application-independent schema with application-specific information; and (3) lineage information obtained by interrogating the patterns of procedure calls, argument values, and metadata inherent in the workflow graphs that describe the indirect nature of the production of a given data object. We describe below the general nature of these three dimensions, and then discuss how queries can be defined that join across these dimensions. We provide detailed query examples drawn from scientific applications in Section 4 and Section 5.

Virtual Data Relationship Queries. The core queries in our model are based on the fundamental entities of the virtual data schema: the prospective declarations of procedure definitions and calls and the retrospective records of actual procedure invocations. These queries focus on the primary tables of the virtual data schema. The following examples illustrate the range of queries that are supported. The first two forms of queries deal with prospective information, while the third deals with retrospective information.

Fundamental Queries of Entity Attributes: Find procedures and calls by namespace, name, and version; find all the calls that invoke a given procedure.

Query by Parameters: Find procedures that pass a specified parameter; find procedure calls that pass a parameter of a specified type and/or value in a specified direction (i.e., input or output); find invocations that executed with a specified parameter value and direction; find procedure calls that process a specified file as input or output; find all files consumed or returned by a specified procedure call.

Query of Invocation Records: find invocation records by procedure or procedure-call namespace, name, version; find procedures or procedure calls executed at a specified site; find procedures or calls executed at a specified host; find invocations run on machines with a specified OS type; find jobs with a specified exit status; find jobs with run time > a specified *r*; find jobs within a set of jobs that ran longer than twice that the set's average time; find invocations that produce files of a specified type and size; find the invocation records that produce or consume a specified dataset.

Annotation Queries. The annotation capabilities provided by the virtual data model on procedures, arguments, calls, datasets, and workflows form the basis for the second query dimension. While various applications may use these annotations to maintain application-specific provenance, we consider this a separate dimension from the provenance information that is intrinsic to the virtual data model.

Annotation queries can, for example, select all annotations for any annotatable virtual data object or set of objects, or select from an annotation result set based on any of subject, predicate, object, object type, user, or annotation date.

Annotations can also be used to select virtual data objects: for example, find all objects (of any type) annotated with predicate *p* of type *t* and value *v*; objects of a specific type annotated with predicate *p* of type *t* and value *v*; or objects (one type or any type) annotated by same set of attribute predicates.

Lineage Graph Queries. A powerful source of information in a virtual data system is the lineage relationships [WS97] that we can derive for all data products. For example, knowing that the inputs to a procedure *Ak* were processed by *Ai* can often tell a scientist important characteristics about the results that *Ak* will derive. Knowing further whether *Aj* processed the output of *Ai* somewhere between those two steps may determine whether further analysis of that chain of data is required.

A simple class of lineage graph queries refers to information that has been propagated along derivation relationships. For example, "find datasets derived from dataset *d*" or "find ancestor datasets to dataset *d* that have type *t*."

More complex queries may refer to patterns within the derivation graph. Much work has been done in this field: for example, Giugno and Shasha [GS02] describe a model for such patterns in a system called GraphGrep. We propose to adapt the GraphGrep model here to the specific problem of matching workflow graphs. We sketch here how we expect to apply this model to enable pattern-based searches of derivation graphs.

The basic approach is to introduce special objects that can match specific patterns of procedures, calls, and invocations, enabling the composition of "workpattern" objects that can perform powerful searches and queries on the workflows in our database. The semantics of such matches work as follows. Procedure patterns, call patterns, and invocation patterns, chained into a DAG within a workpattern object, can match either fixed or varying numbers of nodes of their corresponding object types in any workflow defined in the database. The nodes of a workpattern graph can

match procedure definitions or calls that meet criteria such as argument name, argument values, argument types, and/or annotations.

Performing a query on a workpattern can select a set of workflows, where in each selected workflow one or more subgraphs are matched. The target search space of a workpattern query can be either the entire database, or a specific workflow or set of workflows selected through a prior search. Using the query model defined above, we can perform queries such as: find datasets that were derived within *N* levels of procedure *p*; find datasets that are the result of workpattern *wp*; and find the procedure calls in workflow *w* whose inputs have been processed by any workflow matching workpattern *wp*.

Provenance Queries in Multiple Dimensions. The capabilities of the queries defined above are amplified by the ability to join them flexibly across multiple dimensions of the virtual data schema. For example, we may ask for procedures with a specified signature that have been called with specific argument values (or ranges) and that match an annotation query; the metadata values for a specified set of predicates from a list returned by another query; or the minimum, maximum, and average run times of a set of procedure calls matching workpattern *wp* and annotation query *q*.

For example, a set of procedures selected by a workpattern query can be used to select metadata values that are then used as a search key to select a set of procedure calls. This level of nesting can be used to successively filter (or expand) a result set, and such query chaining can take place to effectively arbitrary depths (limited only by the capabilities of the underlying database system).

Modification and Composition Queries. Maintaining dataset, procedure, workflow, annotation, and provenance information in an integrated schema facilitates not only powerful queries, but also the ability to couple queries with database update procedures to define new procedures, annotations, and work requests. We illustrate such possibilities below.

Change Arguments: For every procedure call *p1* to procedures in namespace *n* with annotation *m*, create a new procedure *p2* with argument *a* replaced by an expression *e*.

Change Procedures: In every workflow *w* matching workpattern *wp*, create a new workflow with the same name but a new version number in which procedure *p1* is changed into procedure *p2* (which must have the same signature).

Edit Subgraphs of a Workflow: In every workflow *w* matching workpattern *wp*, create a new workflow with the same name but a new version number in which the matching workpattern subgraph is changed to a specified new workflow subgraph. (The supplied replacement workflow subgraph must have the same signature.)

Replicate a Workflow: Given a workflow *w* to replicate, for each procedure *p2* returned by query *Q*, create a new workflow *w2(p2)* by replacing occurrences of *p* in *w* with *p2*. Each *p2* returned by *Q* must have the same signature as *p*.

Edit Metadata: In every workflow *w* matching workpattern *wp*, edit annotations on datasets output by the subgraph matched by *wp*, changing the value of metadata predicate *p* to a new value *nv*.

4 Query Examples Drawn from fMRI Science Use Cases

The capabilities that we have described are only interesting if they provide utility to users addressing real data analysis problems. In this section we show such use cases, drawing examples from the field of functional MRI research [HSM02], in which MRI brain images of some subjects are firstly spatially aligned, and then averaged to produce a single image. The procedures employ the AIR (automated image registration) suite [WG+98a,WG+98b] to create an averaged brain from a collection of high resolution anatomical data. The images are annotated with metadata tags such as *studyModality*, *center*, and *state*. We use the categorization of queries introduced earlier in Section 3.

Virtual Data Relationship Queries

- Find all the procedures in namespace */pub/bin/std* that have inputs of type *SubjectImage* and outputs of type *ThumbNailImage*.
- Find all *alignlinear* calls (including all arguments), in XML format, with argument *model=rigid*, and that generated more than 10,000 page faults, on *ia64* processors.
- Find all calls to procedure *alignlinear*, and their runtimes**,** with argument *model=rigid* that ran in less than 30 minutes on non-*ia64* processors.
- Find the average runtime of all *alignlinear* calls with argument *model=rigid* that ran in less than 30 minutes.
- Find all procedure calls within workflow */prod/2005/0305/prep* whose inputs were linearly aligned with *model=affine*

Annotation Queries

- Find all the datasets that have metadata annotation *studyModality* with values *speech*, *visual* or *audio*. Show all the annotation tags of this set of datasets.
- Show the values of all annotation predicates *developerName* of procedures that accept or produce an argument of type *Study* with predicate *studyModality =audio*.

Lineage Queries

- Given the workpattern:
  alignlinear (model=affine) ⇨ *reslice (axis=x, intensify=3)* ⇨ *softmean*
  find all output datasets produced by *softmean* calls that were linear-aligned with *model=affine*. (I.e., "where *softmean* was preceded in the workflow, directly or indirectly, by an *alignlinear* call with argument *model=affine*")
- Find all output datasets of *softmean* that were resliced with *intensify=3*. (Here we want a *softmean* that is directly preceded by the requested pattern.)

Combined Queries

- Find procedures that take an *ImageAtlas* dataset and a *Date* as arguments, have been called with dataset *atlas.std.2005.img*, and have annotation *QALevel* with value > 5.6.
- Find all metadata tags *studyModality* on result datasets that were linearly aligned with parameter *model=affine* and with an input dataset annotated with *center* set to *UChicago*.

- Find the output dataset names (and all their metadata tags) that were linearly aligned with *model=affine* and with input file metadata *center=UChicago*.
- Find all the metadata tags *center* with values in the set (*UIUC, UChicago, UIC*) of output datasets of *softmean*.
- Find all the metadata tags *center* with values in set (*UIUC, UChicago, UIC*) of outputs of *softmean* that were aligned with *model=affine.*
- Find all the metadata tags *studyModality* on results of softmean that were linearly aligned with *model= affine*, and whose output datasets have annotation *state = IL.*

As these examples show, our provenance architecture allows for the expression of a wide range of interesting queries. We need to conduct further experimentation with additional applications, larger and more diverse user communities, and larger data collections to verify that we can both pose and answer efficiently the questions that users want to ask in practice.

5 Implementation and Experience

The VDS implementation of virtual data mechanisms allows for declarative specification of data, procedures, computations and their relationships, using VDL. VDL definitions and provenance data are stored in a "virtual data catalog" (VDC), typically implemented as a relational database and accessed via SQL. We employ an adapter layer to allow the use of different relational database implementations, and support the use of XML databases for the VDC. The actual physical schema used in our implementation is slightly more complex than the logical-level model shown in Figure 1, but captures essentially the same information. The graph structure of workflow objects is currently maintained in an external XML document. VDC queries are parsed and translated into SQL or XQuery/XPath statements to apply against the VDC database.

The physical schema uses a separate value table for each of the five metadata value types supported (string, int, float, date, boolean). This approach enables us to utilize native database searches that treat the data type of the object properly and efficiently (e.g., proper comparison and collation for floating point numbers and dates).

VDS translates requests to derive virtual data products into workflows that may execute at multiple distributed locations. Runtime provenance is obtained by executing VDL procedures under a uniform parent-process wrapper that collects information about the execution of the child application, and its derived files, using OS services. This information is then routed back to the workflow enactment engine via embedded steps in the workflow and saved in the virtual data catalog as invocation records.

An example of the use of virtual data provenance recording is seen in the analysis of provenance information captured by the ATLAS high energy physics experiment to generate simulated events using VDS from 6/2004 through 12/2005. In this period, 20 different simulation procedures were defined in a central US-ATLAS VDC located at Brookhaven National Lab. This virtual data catalog captured 1.2M run-time (retrospective) provenance records, of which 574K described procedure invocations

detailed in the same number of prospective provenance records in the database. 447K unique simulation datasets (logical files) were derived from these invocations.

We can probe the provenance in this catalog with queries that physicists can usefully employ to search for and assess these simulation results. Questions like the following (translated to SQL) can be easily answered (with actual results shown):

Q: List all the procedures that have argument name 'cleanLevel':

```
=> brureconx  evgenx ... G4simulx g4simx  g4simxM pileup testreconx
```

Q: How many jobs running procedures with argument name 'cleanLevel' were run on Linux 2.4.28 kernels?

```
=>   g4digitx   39
     g4simx    340
```

Q: List calls of procedure 'g4simx' with argument eta_min=-5.0 and eta_max=5.0 that were run on 2.4.28 kernels, in Dec 2004?

```
=>  g4simx.CPE_4922_15
    g4simx.CPE_4922_202
    ... (total 285 calls)
```

Simple aggregrations and statistics can also be carried out over the records, for reporting purposes, as in the following examples.

Q: List the total number of jobs run in each month of 2004:

year	month	number_of_jobs
2004	06	1433
2004	07	13331
2004	08	21076
2004	09	20807
2004	10	32364
2004	11	39681
2004	12	14734

Q: List the total run time (in unit of year) of jobs run in each month of 2004:

year	month	run_time(years)
2004	06	0.4
2004	07	20
2004	08	34
2004	09	40
2004	10	15
2004	11	15
2004	12	8.9

Another application of VDS to capture and leverage provenance information is in the QuarkNet nationwide physics education project [BG+05]. In this project, data from cosmic ray detectors located in about 200 high schools in the US uploaded raw data into a data analysis portal driven by VDS. The raw data was annotated and then processed with a set of analysis tools to plot cosmic ray activity under a variety of experimental conditions and derive and document scientific conclusions, modeling closely the processes used in experimental physics collaborations. In this application trial 108 different procedures were used to process 6,330 files (total raw and derived) and to annotate them with 134,834 metadata tuples. A sample query of the

annotations on a data file (a flux study result derived from data gathered by detector 180 channel 1 on 07/30/2004) yields:

```
project:      cosmic          city:          Batavia
group:        fermigroup      state:         IL
study:        flux            creationdate:  2005-01-13 17:44:20.512
detectorid:   180             rawdate:       2004-07-30 19:42:57.0
```

A query to select datasets based on annotations, such as "find all the *blessed* data from *Fermilab*" is expressed in SQL as:

```
select name from anno_lfn f, anno_bool b where f.mkey='blessed' and
b.value=true and f.id=b.id intersect select name from anno_lfn f2,
anno_text t where f2.mkey='school' and t.value='Fermilab'
and f2.id=t.id
```

which returns:

```
180.2004.0730.35
...
999.2005.0604.0
(total 5 rows)
```

6 Related Work

Work on provenance in database systems has focused on determining the source data (tuples) used to produce an item. Cui and Widom [CW00, CWW00] record the relational queries used to construct materialized views in a data warehouse, and then exploit this information to find the source data that contributed to the given data item. Buneman et al. [BKT01] distinguish between *why-provenance* and *where-provenance*. The former explains why a piece of data is in the database, i.e., what data sets (tuples) contributed to a data item, while the latter keeps track of the location of a data item in its source. The mutability of database tables and records poses significant challenges. In contrast, we address provenance issues within the context of data analyses performed using programs that are assumed not to modify their input datasets. In this context, we can go beyond *why* and *where* to address issues of *how* a data product was (or can be) derived, *what* are the procedure definitions and annotations, and to *which* workflow the procedure belongs.

Various systems support provenance tracking in the scientific community. SAM [MC+03] has a "laboratory notebook" model of provenance tracking in which metadata can be added to data items stored in a repository. However, SAM does not define the format or schema of the metadata or an underlying computational model. The same is true of other notebook schemes [BK+06]. In myGrid, documentation about workflow execution is recorded and stored in a user's personal repository, along with other metadata [ZG+03,ZGR06], to support personalized provenance tracking of bioinformatics services and workflows.

Szomszor and Moreau [SM03] propose a service-based architecture for recording provenance in a Grid environment. They rely on a workflow enactment engine to submit service invocation information to a provenance service. Moreau et al. [GM+05] describe an implementation-independent architecture for provenance systems. They describe a logical architecture in which so-called *p-assertions* can be

submitted and retrieved from a *p-store* by various actors, and a process architecture for system security and distribution. This system has been applied to physics [BM06], engineering [KS06], and transplant management [AV+06], among others.

Our approach is distinguished from that of Moreau et al. by its integration of provenance with a particular computational model, namely that captured by our functional virtual data language. This model makes it feasible for us to capture high-fidelity retrospective and prospective provenance information, and then to interpret and query this information in powerful ways. In particular, our focus on a specific computational models means that we can define a specific schema that maintains information about such constructs as "procedures," "calls," and "workflows," in addition to general purpose assertions. The two schemas can in fact converge: the Moreau schema can subsume the information we describe here, and we can integrate the information of the Moreau schema (in addition to our custom-tailored virtual data schema) by using a more general RDF model for our metadata annotations. With such a schema, metadata annotations can be interpreted broadly, and any annotation can be associated with our core data (file) and executable (procedure, workflow) objects.

PASS [MH+06, BG+06] is a storage system that automatically collects and maintains provenance information. In comparison with VDS, PASS discovers the components and environments required for the production of a specific data item by tracking system calls, where in VDS, we rely on explicit declarations of such dependencies. However, the two systems are complimentary in many ways: PASS can help discover missing pieces in our "declared" provenance, for instance, a data item that is necessary for a derivation process, or an extra data product produced during the process that was not captured in the procedure definition. On the other hand, PASS cannot track complete provenance beyond local file systems, such as in a Grid environment, without provenance-aware applications and environments. Both systems also share similar graph traversal mechanisms to build the ancestry tree for a data product, and have the same set of requirements for provenance and workpattern queries; they also face similar challenges determining the granularity at which the systems should capture provenance information, and the lifetime management of the captured information. PASS has the goal of generalizing the production of certain individual items into a workflow-like pattern, for instance, running "*sort a > b"* would involve the same set of operations for any file produced in such a process; where in VDS we can discover such patterns in an interactive environment such as the Chiron virtual data portal [ZW+05], as users tend to repeat the same derivation process for a set of data items. Similar comments can be made about the automated provenance recording techniques being developed by Barja and Digiampietri within the context of Microsoft's Windows Workflow Foundation [BD06].

The evolution or provenance of a workflow itself is also vital in scientific analysis. VisTrails [CF+06, FS+06] captures the notion of an evolving dataflow, and implements a history management mechanism to maintain versions of a dataflow, thus allowing a scientist to return to previous steps, apply a dataflow instance to different input data, explore the parameter space of the dataflow, and (while performing these steps) compare the associated visualization results. In VDS, workflows are also composed of individual steps (procedure calls), and chained together by data dependencies. To provide similar functionality, we can either add a versioning mechanism to our system, or maintain workflow contruction-specific

metadata annotations to track the history of a workflow. The modification and composition query capabilities described in Section 3 support parameter exploration and comparative analysis. Workpattern queries should allow flexible and powerful workflow editing and transformation.

7 Conclusion and Future Directions

We have described the representation and query of both prospective and retrospective provenance information in our virtual data provenance model, presented examples of how provenance can be employed in representative science processes (in neuroscience and physics), and shown how powerful queries can be used to derive valuable knowledge in data analysis processes. These queries select on various combinations of procedure information, data, metadata, and (what seems particularly interesting) workflow patterns.

While the model described here is based on application programs rather than Web services, we believe the same model of provenance applies equally well in a service oriented architecture, with no loss of generality.

Future extensions to the virtual data provenance model include maintaining a transactional provenance trail of changes to metadata annotations, studies of scalability, management of provenance data retention, and the application of the model to a distributed web of provenance catalogs employing a similar schema (see, for example, AstroDAS [BMP06]). We are also eager to perform more comprehensive usability experiments with a wide range of users.

Acknowledgments

This work was supported in part by the National Science Foundation GriPhyN project under contract ITR-086044 and by the U.S. Department of Energy under contract W-31-109-Eng-38. We acknowledge the contributions to this work of Jens Voeckler, Jed Dobson, and members of the QuarkNet project.

References

[AV+06] Alvarez, S., Vazquez-Salceda, J., Kifor, T., Varga, L.Z. and Willmott, S. Applying Provenance in Distributed Organ Transplant Management, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS, 38-47.

[BG+05] Bardeen, M., Gilbert, E., Jordan, T., Nepywoda, P., Quigg, E., Wilde, M. and Zhao, Y. The QuarkNet/grid collaborative learning e-Lab. IEEE International Symposium on Cluster Computing and the Grid, 2005. CCGrid 2005. Vol. 1 pp. 27-34. 9 May 2005. DOI 10.1109/CCGRID.2005.1558530.

[BD06] Barga, R.S. and Digiampietri, L.A. Automatic Generation of Workflow Provenance, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS, 1-9.

[BG+06] Braun, U., Garfinkel, S., Holland, D., Muniswamy-Reddy, K. and Seltzer, M. Issues in Automatic Provenance Collection, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS, 132-144.

[BK+06] Bourilkov, D., Khandelwal, V., Kulkarni, A. and Totala, S. Virtual Logbooks and Collaboration in Science and Software Development, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS, 19-27.

[BKT01] Buneman, P., Khanna, S., and Tan. W.-C. Why and Where: A Characterization of Data Provenance. In International Conference on Database Theory, 2001.

[BM06] Branco, M. and Moreau, L. Enabling provenance on large scale e-Science applications, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS, 55-63.

[BMP06] Bose, R., Mann, R.G. and Prina-Ricotti, D. AstroDAS: Sharing Assertions across Astronomy Catalogues through Distributed Annotation, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS, 154-163.

[CF+06] Callahan, S.P., Freire, J., Santos, E., Scheidegger, C.E., Silva C.T. and Vo, H.T. Managing the Evolution of Dataflows with VisTrails. IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow) 2006.

[CW00] Cui, Y. and Widom, J., Practical Lineage Tracing in Data Warehouses. In 16th International Conference on Data Engineering, (2000), 367–378.

[CWW00] Cui, Y., Widom, J. and Wiener, J.L. Tracing the Lineage of View Data in a Warehousing Environment. ACM Transactions on Database Systems, 25 (2). 179–227. 2000.

[DS+05] Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J.C. and Katz, D.S. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. Scientific Programming, 13 (3). 219-237. 2005.

[F05] Foster, I., Globus Toolkit Version 4: Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing, 2005, Springer-Verlag LNCS 3779, 2-13.

[FS+06] Freire, J., Silva, C.T., Callahan, S.P., Santos, E., Scheidegger, C.E. and Vo, H.T. Managing Rapidly-Evolving Scientific Workflows, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS, 10-19,

[FV+02] Foster, I., Voeckler, J., Wilde, M. and Zhao, Y. Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation. in 14th Conference on Scientific and Statistical Database Management, (2002).

[FT+02] Frey, J., Tannenbaum, T., Foster, I., Livny, M. and Tuecke, S. Condor-G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing, 5 (3). 237-246. 2002.

[GM+05] Groth, P., Miles, S., Tan, V. and Moreau L. Architecture for Provenance Systems. Technical report, University of Southampton, October 2005.

[GS02] Giugno, R. and Shasha, D. Graphgrep: A fast and universal method for querying graphs. In Proceeding of the IEEE International Conference in Pattern recognition (ICPR), Quebec, Canada, August 2002.

[KS06] Kloss, G.K. and Schreiber, A. Provenance Implementation in a Scientific Simulation Environment, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS, 28-37.

[HSM04] Huettel, S., Song, A. and McCarthy, G. Functional Magnetic Resonance Imaging. Sinauer Associates, 2004.

[MC+03] Myers, J.D., Chappell, A.R., Elder, M., Geist, A. and Schwidder, J. Re-integrating the research record. IEEE Computing in Science & Engineering, pages 44–50, 2003.

[MH+06] Muniswamy-Reddy, K., Holland, D., Braun, U. and Seltzer, M. Provenance-Aware Storage Systems, 2006 USENIX Annual Technical Conference, Boston, MA, June 2006.

[SKD06] Singh, G., Kesselman, C. and Deelman, E. Optimizing Grid-Based Workflow Execution. Journal of Grid Computing, 3 (3-4). 201-219. 2006.

[SM03] Szomszor, M. and Moreau, L. Recording and reasoning over data provenance in web and grid services. In Intl. Conf. on Ontologies, Databases and Applications of Semantics, LNCS 2888, 2003.

[WG+98a] Woods, R.P., Grafton, S.T., Holmes, C.J., Cherry, S.R. and Mazziotta, J.C. Automated image registration: I. General methods and intrasubject, intramodality validation. Journal of Computer Assisted Tomography 1998;22:139-152.

[WG+98b] Woods, R.P., Grafton, S.T., Watson, J.D.G, Sicotte, N.L. and Mazziotta, J.C. Automated image registration: II. Intersubject validation of linear and nonlinear models. Journal of Computer Assisted Tomography 1998;22:153-165.

[WS97] Woodruff, A. and Stonebraker, M., Supporting Fine-Grained Data Lineage in a Database Visualization Environment. 13th International Conference on Data Engineering, 1997, 91-102.

[ZG+03] Zhao, J., Goble, C., Greenwood, M., Wroe, C. and Stevens, R. Annotating, linking and browsing provenance logs for e-science. In Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, October 2003.

[ZGR06] Zhao, J. Goble, C. and Stevens, R. An Identity Crisis in the Life Sciences, International Provenance and Annotation Workshop (I-PAW), 2006, Springer-Verlag LNCS.

[ZD+05] Zhao, Y., Dobson, J., Foster, I., Moreau, L. and Wilde, M. A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data. SIGMOD Record, 34 (3). 37-43. 2005.

[ZW+05] Zhao, Y., Wilde, M., Foster, I., Voeckler, J., Dobson, J., Gilbert, E., Jordan, T. and Quigg, E. Virtual Data Grid Middleware Services for Data-Intensive Science. Concurrency and Computation: Practice and Experience, DOI: 10.1002/cpe.968, 2005.