Applying the Virtual Data Provenance Model

Yong Zhao, University of Chicago, yongzh@cs.uchicago.edu
Michael Wilde, University of Chicago and Argonne National Laboratory
Ian Foster, University of Chicago and Argonne National Laboratory

In many domains of science, engineering, and commerce, data analysis systems are employed to derive knowledge from datasets describing experimental results or simulated phenomena. To support such analyses, we have developed a "virtual data system" in which a uniform notation is used to request the invocation of data transformation procedures and to record how every result derived by the system was produced. We maintain such prospective and retrospective information in an integrated schema alongside semantic annotations, and thus enable 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 provides, and examples of how these capabilities can serve the scientific process.

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 procedure that was used, or can be used, to produce or reproduce it. Such associations are made 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, by "procedure" we refer to executable application programs, but the paradigm applies equally well to a service-oriented procedure model).

We view provenance in this context as being of two parts: all the aspects of the procedure or workflow for creating a data object (*prospective* provenance, or "recipe") as well as information about the runtime environment in which these procedures were executed and the resources used in their invocations (*retrospective* provenance).

While only the prospective information is needed to produce or reproduce a data object, the *complete* provenance record—prospective and retrospective—provides a more complete understanding of the data. 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 pose new questions.

The Virtual Data System that we have developed to implement this model [ZW+05] maintains a precise record of data transformation procedures, inputs (both data and parameter settings) to transformations, the environment in which transformations were invoked, and relevant data about how a transformation behaved (e.g., duration). Armed with this information, we can track, for any data object created within the system, a derivation history recursively back to raw input data, and thus obtain accurate 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 recipes for conducting new investigations that build on previous findings and approaches.

We focus in this paper on illustrating the virtual data approach to integrating prospective and retrospective provenance with semantic annotations; on the powerful queries that can be performed on such an integrated base; and on 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 and queries 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.

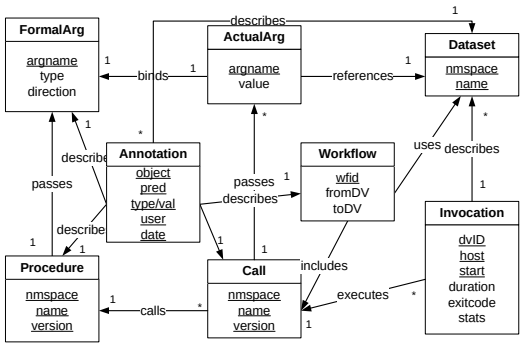

Figure 1: Schema for provenance and annotation

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, one for each edge of the workflow graph. 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.

The integration of workflows here enables queries to consider the provenance history of data objects, and the interrelationships between procedures based on the patterns in which they are actually used in defined workflows.

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*."

We find it useful to think of the virtual data query model as having three major dimensions (as shown below): 1) direct provenance from the 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

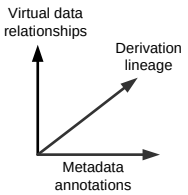

Fig 2:Query dimensions

production of a given data object. We describe below the general nature of these three dimensions.

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, the retrospective records of actual procedure invocations, and so forth. These queries focus on the primary tables of the virtual data schema. 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 parameter named *p*; find procedure calls that pass a parameter of type *t* with value *v* in direction (i.e., input or output) *d*; find invocations that executed with parameter *p* of value *v* and direction *d*; find procedure calls that process logical file *f* in direction *d* (i.e. as input or output); find all the files of direction *d* consumed or returned by a procedure call.

Query of invocation records: find invocation records by procedure or procedure-call namespace, name, version; find procedures or procedure calls executed at site name *sn*; find procedures or calls executed at host *h*; find invocations run on machines with OS type *os*; find jobs with exit status *s*; find jobs with run time > *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 type *t* with size < *s*; find the invocation records that produce or consume a dataset *d*.

Annotation queries. The annotation capabilities provided by the virtual data model on procedures, arguments, datasets, and workflows form the basis for this next dimension of query. 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. (As we discuss at the end of this section, users may form queries that join across these dimensions).

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 be used to select virtual data objects as well: 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 that we can derive for all data products. For example, knowing that the inputs to an application *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 refer 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 adapt the GraphGrep model here to the specific problem of matching workflow graphs.

We provide this capability through 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. Transformation 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 transformations with a specified signature that have been called with specific argument values (or ranges) and which match an annotation query; the metadata values for a specified set of predicates from a transformation 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.

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 predicate *status* to *newvalue*.

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 relevant to the field of functional MRI research [HSM02]. 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 which 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:
  align (model=affine) ⇨ *reslice (axis=x, intensify=3)* ⇨ *softmean*
  find all output datasets of 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 *study-type* on result datasets that were linearly aligned with parameter *model=affine* and with an input dataset annotated with *center* set to *UofChicago*.
- Find the output dataset names (*and all their metadata tags)* that were linearly aligned with *model=affine* and with input LFN metadata *center=UChicago*.
- Find all the metadata tags *school* with values in the set (*UIUC, UChicago, UIC*) of output datasets of *softmean*.
- Find all the metadata tags *school* with values in set (*UIUC, UChicago, UIC*) of outputs of *softmean* that were aligned with *model=affine.*
- Find all the metadata tags *study* on results of softmean that were linearly aligned with model= affine, and whose output datasets have annotation *state = IL*.

5 Implementation and Experience

The VDS implementation of virtual data mechanisms allows for declarative specification of data, procedures, computations and their relationships, using a Virtual data Language, VDL [FV+02]. VDL definitions and provenance data is 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 complete set of queries described here – notably those for lineage-graph-matching multi-dimension processing, and annotation date/owner – are still under development.

For metadata, the physical schema uses a separate value table for each of the 5 metadata value types supported (string, int, float, date, boolean). This 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).

Requests to derive virtual data products are mapped 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)
```

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: `80.2004.0730.35 ... 999.2005.0604.0`

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, and 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.

A few 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. In myGrid, documentation about workflow execution is recorded and stored in a user's personal repository, along with other metadata [ZG+03], 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. In [GM+05], Moreau et al describes an implementation independent architecture for provenance systems. The paper covers the logical architecture where p-assertions can be submitted and retrieved from a p-store by various actors, and the process architecture for system security and distribution.

One major difference of our system is that we support retrospective and prospective provenance, which enables powerful queries and composition capabilities, and high fidelity (while practical) recording. In contrast to Moreau et al, we have a specific schema that keeps the "backbone" of what we feel is the most critical part of provenance information, 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 (logical file) and executable (transformation, workflow) objects. The contribution of our work is the identification of a specific schema that permits query and composition on the essential aspects of how data objects were derived in a long-running analysis process of arbitrary duration and complexity, the integration of multiple dimensions of provenance information into a unified, practical framework that provides useful answers regarding the data analysis process, and the utilization of graph queries to harvest the knowledge implicit in lineage information.

7 Conclusion and Future Directions

We have shown clear and practical examples of how provenance can be employed in a few representative science processes (in neuroscience and physics) and how powerful queries can be utilized to provide information that is highly valuable in the data analysis process.

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.

8 References

[BG+05] M. Bardeen, E. Gilbert, T. Jordan, P. Nepywoda, E. Quigg, M. Wilde, Y. Zhao .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

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

[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.

[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).

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

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

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

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

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

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

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