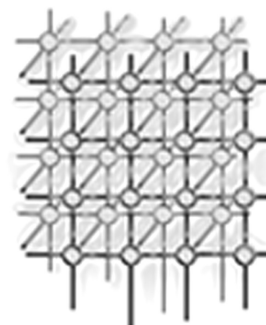

Query capabilities of the Karma provenance framework

Yogesh L. Simmhan^{*,†}, Beth Plale and Dennis Gannon

Computer Science Department, Indiana University, Bloomington IN 47405, USA.



SUMMARY

Provenance metadata in e-Science captures the derivation history of data products generated from scientific workflows. Provenance forms a glue linking workflow execution with associated data products, and finds use in determining the quality of derived data, tracking resource usage, and for verifying and validating scientific experiments. In this article, we discuss the scope of provenance collected in the Karma provenance framework used in the LEAD Cyberinfrastructure project, distinguishing provenance metadata from generic annotations. We further describe our approaches to querying for different forms of provenance in Karma while addressing the queries proposed in the First Provenance Challenge. We use an incremental, building-block method to construct provenance queries based on the fundamental querying capabilities provided by the Karma service centered on the provenance data model. This has the advantage of keeping the Karma service generic and simple, and yet supports a wide range of queries. Karma successfully answers all but one Challenge query. Copyright © 2007 John Wiley & Sons, Ltd.

KEY WORDS: data and process provenance; scientific workflows; provenance queries

1. INTRODUCTION

Provenance in e-Science [11, 1] is a form of metadata capturing the derivation history of data products generated from executing scientific tasks, often modeled as workflows. Provenance forms the glue linking workflow executions with associated data products, and finds use in determining the quality of derived data, tracking resource usage, verifying and validating scientific experiments, and for different forms information discovery through querying and mining.

The Karma provenance framework [12] is used in the LEAD Cyberinfrastructure [5, 7] project to collect and query over provenance generated from meso-scale weather forecasting workflows. The Karma system was one of the entries in the First Provenance Challenge[10] and this article describes our experiences in addressing the Challenge problems. Karma places a greater emphasis

*Correspondence to: Yogesh L. Simmhan, Computer Science Department, Indiana University, 150 S. Woodlawn Ave., Bloomington IN 47405, USA.

†E-mail: ysimmhan@cs.indiana.edu

Contract/grant sponsor: National Science Foundation; contract/grant number: ATM-0331480



on the collection of provenance from scientific workflow executions as compared to providing rich querying capability. The provenance Challenge provided an opportunity to test different approaches for querying for provenance in Karma and provided an insight on future paths of enhancements to the query interface. Karma also takes a conservative view of the scope of provenance metadata, limiting it largely to the causal chain of events that lead to the creation of data products through the execution of processes. Generic annotations are considered beyond the purview of a provenance system. There are several general purpose metadata catalogs available for various resources on the Grid [4, 14, 13] and the provenance system can play a pivotal role in the integration of information from these information services – without having to be burdened by having to store, manage, and query upon annotations. We had a chance to pit this philosophy against the Challenge queries, several of which deal with queries over annotations, and found our system to answer all but one of the Challenge queries.

The rest of this article is organized as follows: in Section 2, we briefly describe the Karma provenance framework and its data model, followed by a discussion of the query capabilities of Karma and its application to the Challenge problem in Section 3, and we conclude in Section 4 with some comparisons with other provenance systems in the Challenge and our future work.

2. THE KARMA PROVENANCE FRAMEWORK

The Karma provenance framework [12] is used to collect and query over provenance on data products derived from scientific workflows executing in a service oriented architecture. Scientific experiments in the LEAD project are designed as workflows composed out of web services. These services encapsulate specific scientific applications, taking input data and parameters and generating output data, and are usually command-line applications wrapped using our custom Application Factory Toolkit that generates generic web services wrappers for arbitrary scientific tasks [8, 7]. Services used to compose workflows may themselves be other workflows, allowing for a hierarchical composition of workflows. LEAD uses a variation of Business Process Execution Language (BPEL) as the workflow description language and a centralized GPEL workflow engine to execute them on the LEAD Grid [15].

Provenance Activities. Provenance is collected from running workflows by instrumenting the clients and services to generate *provenance activities* as services are invoked [12]. These activities identify the boundaries of execution of each invocation and are generated at the beginning and ending of an invocation by a client of a service, when a data is produced, and when it is consumed. All services, clients, and data products are uniquely identified by a global ID and the granularity of data produced or consumed during an invocation is arbitrary – anything that can be mapped from the global ID of the data back to an identifiable data resource. The state of the service at the point of invocation is captured by its *entity ID*, consisting of the ID of the workflow the service is part of, the ID for the service itself, the logical time in the workflow execution that this invocation takes place at, and the node in the workflow graph that this invocation represents. The client to the service, which may itself be another service, has a similar entity ID defined for it. A combination of the entity IDs for the client and service involved in the invocation makes it possible to uniquely identify the invocation in time and space. These entity IDs and data product IDs are present in the activities that are generated, along with relevant metadata such as timestamps and data product locations. The provenance activities adhere to an XML schema that defines the elements they contain. Activities can also optionally carry generic XML metadata describing it, though this is purely for convenience. Two sample activities generated by



```
<serviceInvoked xmlns="http://lead.extreme.indiana.edu/namespaces/2006/06/workflow_tracking">
<notificationSource workflowNodeID="ConvertService_4" workflowTimestep="36"
workflowID="tag:gpel.leadproject.org,2006:698/ProvenanceChallengeBrainWorkflow17/instance1"
serviceID="urn:qname:http://www.extreme.indiana.edu/karma/challenge06:ConvertService" />
<timestamp>2006-09-10T23:56:28.677Z</timestamp>
<description>Convert Service was Invoked</description>
<request><header>...</header><body>...</body></request>
<initiator
serviceID="tag:gpel.leadproject.org,2006:698/ProvenanceChallengeBrainWorkflow17/instance1" />
</serviceInvoked>
```

(a)

```
<dataProduced xmlns="http://lead.extreme.indiana.edu/namespaces/2006/06/workflow_tracking">
<notificationSource workflowNodeID="ConvertService_4" workflowTimestep="36"
workflowID="tag:gpel.leadproject.org,2006:698/ProvenanceChallengeBrainWorkflow17/instance1"
serviceID="urn:qname:http://www.extreme.indiana.edu/karma/challenge06:ConvertService" />
<timestamp>2006-09-10T23:56:32.324Z</timestamp>
<dataProduct>
<id>lead:uuid:1157946992-atlas-x.gif</id>
<location>
gsiftp://tyr1.cs.indiana.edu/tmp/20060910235628_Convert/outputData/atlas-x.gif</location>
</dataProduct>
</dataProduced>
```

(b)

```
<dataProvenance xmlns="http://extreme.indiana.edu/namespaces/2006/08/karma/xsd"
xmlns:wft="http://lead.extreme.indiana.edu/namespaces/2006/06/workflow_tracking"
dataProductID="lead:uuid:1157946992-atlas-x.gif" timestamp="2006-09-10T23:56:32.324Z"
dataProductLocation="
gsiftp://tyr1.cs.indiana.edu/tmp/20060910235628_Convert/outputData/atlas-x.gif" />
<producedBy status="InvokingService ServiceInvoked InvokingServiceSucceeded
SendingResult ReceivedResult SendingResponseSucceeded"
requestReceiveTime="2006-09-10T23:56:28.677Z"
responseSendTime="2006-09-10T23:56:32.397Z"
invoker.wft.serviceID="
tag:gpel.leadproject.org,2006:698/ProvenanceChallengeBrainWorkflow17/instance1" />
<invokee wft.workflowNodeID="ConvertService_4" wft.workflowTimestep="36" isWorkflow="false"
wft.workflowID="
tag:gpel.leadproject.org,2006:698/ProvenanceChallengeBrainWorkflow17/instance1"
wft.serviceID="
urn:qname:http://www.extreme.indiana.edu/karma/challenge06:ConvertService" />
</producedBy>
<usingData
dataProductID="lead:uuid:1157946967-atlas-x.pgm" timestamp="2006-09-10T23:56:32.180Z"
dataProductLocation="
gsiftp://tyr7.cs.indiana.edu/tmp/20060910235600_Slicer/outputData/atlas-x.pgm" />
</dataProvenance>
```

(c)

Figure 1. (a) A ServiceInvoked activity generated by a service upon invocation. (b) DataProduced activity generated by a service when it creates a new data product. (c) The data provenance for a data product.

an invoked service are shown in Figure 1(a) and (b). While the ServiceInvoked activity is generated at the moment the service is called, the DataProduced activity is generated whenever a data product is created during the invocation.

The activities form an event driven approach to collecting provenance while the data products exchanged between service invocations establishes a data centric causality trail. In LEAD workflows, the clients for the service invocation is the central workflow engine that orchestrates the service invocations. The engine is instrumented to generate client-side provenance activities as the workflow progresses, while the service wrappers generated for the command-line tasks are also automatically instrumented to generate the service and data provenance activities. Karma supports both a notification based model to asynchronously listen to these activities and also provides a direct web service API to submit the provenance activities synchronously.

Provenance Data Model. Karma exports two primary forms of provenance: *data provenance* and *process provenance*, with variations of each [12]. Process provenance refers to the invocation of a certain process or service, along with the inputs to and outputs from the service in the form of data products. Data provenance describes the derivation of a data product, including the service invocation that created this data and the inputs to it. Both these types of provenance are represented as XML documents following a well defined schema. Data products are referred to by their unique data product ID along with a timestamp of their creation or usage and the location of the data product (replica). Service invocations are represented by a combination of the entity IDs of the clients that invoke the service and that of the service itself. They are also known as the invokers and invokees, making an invocation a pair of invoker and invokee.

A typical data provenance for the data product with ID lead:uuid:1157946992-atlas-x.gif is shown in Figure 1(c). This shows the data product being created on 10th September 2006 at 11:56PM UTC at the tyr1 host by the ConvertService running as part of the



ProvenanceChallengeBrainWorkflow17 and invoked by the workflow engine that acts as the client, passing a data product having ID `lead:uuid:1157946967-atlas-x.pgm` as the input.

Since services are frequently invoked in the context of a workflow, a variation of the process provenance for an invocation is the process provenance for all services invocation that were part of a workflow run. This is effectively a workflow trace represented as a directed graph where service invocations as nodes and data products are edges. Since workflows themselves are abstracted as services, this method can be recursively applied to build a hierarchical provenance graph for service and workflow invocations of arbitrary depth that were executed across organizational boundaries. This conveniently presents different levels of abstraction to the user to retrieve process provenance – at the fine-grained level of a service invocation or at the coarser granularity of a workflow. Since workflows can be hierarchically composed, this allows abstraction of process provenance to arbitrary coarseness based on the workflow design.

In addition to the *immediate provenance* described earlier, data provenance can be recursively tracked beyond the service invocation creating it. Since other data products were used as inputs to the invocation, the immediate provenance for those ancestral data products can be attached to the data provenance to recursively render a *deep provenance* tree going back in time. A corollary of this data provenance is a *usage trail* for the data. This returns the service invocations that use a data product and can be used to go forward in time along the data provenance graph.

Backend Database Model. Provenance activities that arrive as XML documents at the provenance service are decomposed into their primary fields and stored in a relational database. Shredding an XML format into a relational model allows convenient querying over the provenance and allows it to later be disseminated in any representation. The relational schema, outlined in Figure 2, stores the various facets of provenance: services and clients, known as entities; details about services which can also be workflows; invocations of the services by clients that may themselves be services or workflows; data products and their relationship to entities that consume and produce them; and finally the raw provenance activities, called notifications, for provenance about the provenance data itself. In addition to standard fields such as timestamp and unique IDs, the data product, the service, and the notification tables allow generic XML metadata to be recorded for convenience and these can be retrieved as part of provenance for post processing. However, only free-text querying is possible upon them and no syntactic structure is pre-supposed.

The information in these tables can be queried and combined to build different forms of provenance. The basic queries supported by the provenance service API are based on the unique IDs of the data products, services, workflows, and invocations. These are internally translated to SQL queries on the different tables and the relevant tuples extracted to populate the provenance data model. For example, a query for the data provenance of data product with ID `lead:uuid:1157946992-atlas-x.gif` would create a document similar to Figure 1(c), generated by joining information from the data product and data produced tables to identify the data product, the entity and invocation state tables to locate the service and invocation that created this data product, and the data consumed table for invocations that have used this data product. Karma uses the MySQL relational database as the backend implementation.

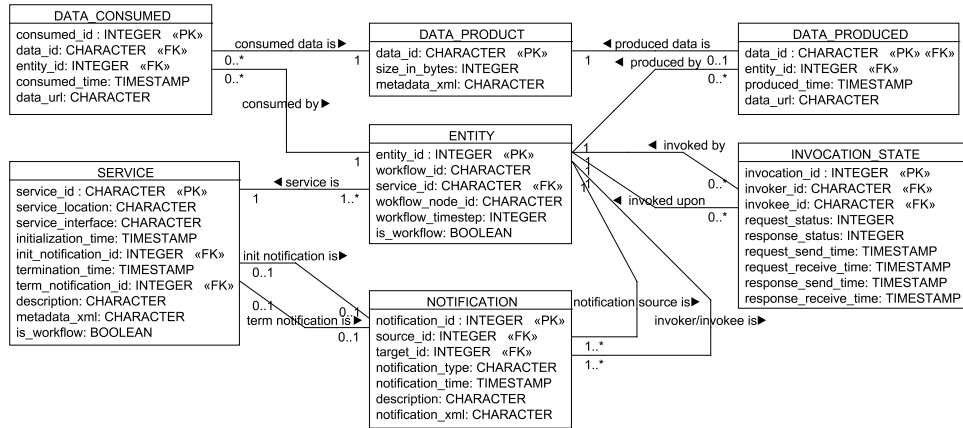


Figure 2. Relational Data Model for Provenance Activities. The Entity table captures information about the source or target of an invocation, the Invocation table relates the source and target of an invocation with the status of the invocation, and the Service table stores details about individual service entities. Between them, they record the *process provenance*. Data Product, Data Produced, and Data Consumed tables maintain details about data created and used by each entity, and record the *data provenance*. The raw provenance activities and annotations are available through the Notification table.

3. QUERY CAPABILITIES

The Provenance Challenge gives a unique chance to showcase the querying capabilities of Karma. The Challenge workflow, described more fully in the introduction to this issue [10], is composed as a BPEL workflow [8, 15] using the XBay Workflow Composer and executed using the GPEL workflow engine on a local cluster. Each process in the workflow is a web service wrapping a shell script using the Application Factory Toolkit, and the shell script invokes the corresponding command-line executable it is mapped to, such as *AlignWarp* or *Softmean*. The services run on different hosts, and all services and data products are assigned unique IDs when created. The web service wrapper automatically registers the services with the Resource Catalog service registry, stages files between hosts using third-party file transfers, registers the data product replicas with a naming service, and generates provenance activities as asynchronous notifications published to a notification broker. The Karma provenance service subscribes to all provenance activities from the workflow run, stores them in the relational database, and makes them available for querying.

The queries defined for the Challenge [10] can be grouped into those based purely on the structure of provenance and those requiring the additional inspection of annotations. Provenance structure queries pertain to information such as “*which process created a data product*”, “*when it was created*”, “*what were the input data products to a process*”, and these applied recursively to ancestral data products. Annotation queries are over additional metadata submitted beyond the scope of provenance that can be generic in nature, such as parameters and attributes describing the deriving process or the data products such as `center=UChicago` describing the location of the data product creator. In the Challenge queries, Q1, Q2, Q3, and Q8 are provenance structure queries, while Q4, Q5, Q6, Q7, and Q9 rely on annotations also.



Table I. Approaches used to perform Challenge queries. Those answered using the service API are marked API; those requiring post-processing by client are under Client; those answered using direct SQL query on database are marked SQL; while the one that was not answered is marked Unsupported.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
API	Client	API	SQL	SQL	SQL	Client	SQL	Unsupported

Three different techniques are used to answer the Challenge queries. Some of the queries on the provenance structure can be directly answered by the provenance query API exposed by the Karma service. Other provenance structure queries require the client to do incremental queries by processing the results of an initial service API query and performing additional service queries based on its results. Finally, queries that involve annotations are directly written as SQL queries over the backend relational database. The approach used to answer each of the Challenge queries is given in Table I. In the ensuing sections, we provide one example for each of these approaches from the Challenge queries. Other Challenge queries that fall in the same category are solved in a similar manner and are not discussed in the interests of brevity. All workflows, services, queries performed, and their results are available online at the Provenance Challenge website [6].

Directly Answered Queries. Queries Q1 and Q3 can be directly answered using the Karma service API. The service API supports several methods to retrieve different forms of provenance metadata described in the data model using the global IDs of workflows, service invocations, and data products. The service API is intentionally simple, presenting the methods `getProcessProvenance`, `getWorkflowTrace`, `getDataProvenance`, `getRecursiveDataProvenance`, and `getDataUsage`. The process provenance and workflow trail both return process provenance metadata given the invocation ID for a service or workflow invocation. While the latter returns the process provenance for all services that were part of the workflow, the former returns this information only for a single service invocation. The workflow trail also optionally does a recursive retrieval of the process provenance of other child workflows invoked as part of this workflow, and the depth of the recursion is configurable. The data provenance can be retrieved for a given data product ID and the recursive variant works backwards in time, fetching the data provenance for the inputs to the process generating this data product and so on. Again, the depth of recursion is parameterized. Finally, the data usage for a data product ID works forward in time, locating the processes that use a certain data product after it is created.

Query Q1 is to *find the process that led to Atlas X Graphic*. This requires us to generate the complete data provenance for the Atlas X Graphic file recursively. This is readily achieved by using the `getRecursiveDataProvenance` method with the depth of recursion set to unlimited. This works backwards from the Atlas X Graphic file and returns all processes and their input and output data products that went to create the Atlas X Graphic file. A Java library is provided to marshal the SOAP document for the web service call and the returned provenance metadata is mapped to XML Bean Java objects for convenience. The method call on that library to get the recursive data provenance for the Atlas X Graphic file, whose data product ID is `lead:uuid:1157946992-atlas-x.gif` is given below, where the `-1` parameter signifies an unlimited recursion depth.



```
RecursiveDataProvenanceType dataProvenance = karmaStub.  
getRecursiveDataProvenance("lead:uuid:1157946992-atlas-x.gif", -1);
```

Similarly, query Q3 to get the data provenance for the Atlas X Graphic file restricted to 3 levels back can be answered by the same method above, replacing the -1 for recursion depth by 3.

Queries Requiring Client-side processing. In order to answer more complex queries than cannot be retrieved through a single service API call, clients use an incremental building-block approach to fetch the necessary provenance information from the Karma service. Queries Q2 and Q7 in the Challenge fall under this category. Query Q2 is similar to Q3 discussed previously and retrieves the recursive data provenance for the Atlas X Graphic file. However, instead of putting a limit on the data provenance based on the depth of recursion, the limit is imposed on a process that is encountered as part of the data provenance history, in this case, **Softmean**. Since this is not directly supported by the service API, we use the `getDataProvenance` API to get the immediate data provenance for the Atlas X Graphic file, and recursively call the `getDataProvenance` method on each input to the deriving processes, moving backwards in the provenance tree until the **Softmean** process is reached. The pseudo-code for the client looks as follows:

```
DataProvenance[] dataProvenance = RecursiveDataProvenanceUntil(  
    'lead:uuid:1157946992-atlas-x.gif', 'urn:qname:...:SoftmeanService')  
  
function RecursiveDataProvenanceUntil(DataProductID dataProductID,  
    URI processID) :: DataProvenance[]  
1. $resultList[] = {}  
2. $dataList[] = {dataProductID}  
3. while ($dataList.size != 0)  
    a. $dataProvenance = karmaStub.getDataProvenance($dataList[0])  
    b. $resultList.add($dataProvenance)  
    c. $dataList.remove(0)  
    d. if ($dataProvenance.producedBy == processID) break  
    e. foreach ($inputData in $dataProvenance.usingData) do  
        i. $dataList.add($inputData)  
4. return $resultList;
```

Query Q7 to find the differences between two workflow runs with slightly different services also requires post-processing by the client. This time, the workflow trace for both workflow executions can be retrieved and then the two graphs can be recursively compared by the client to find the differing processes. Standard algorithms exist to do a `diff` on two graphs, which are what the workflow traces are, and one such algorithm is used to solve this. The differences that can be found are the variations in processes, the differences in input and output data created, the changes in invocation and creation times, and other details available as part of the process provenance data model.

Queries Requiring Access to Backend Database. The Karma provenance service is primarily intended as a provenance recording and querying system, and has limited support for recording generic metadata and annotations. Provenance activities can have annotations and some activities use these to store SOAP request and response messages that were exchanged by service and client during an invocation. All of these are stored in the relational backend and be optionally returned as part of the provenance data model. Karma does not support queries over annotations as part of the service API. However, they can be queried upon by directly executing SQL queries over the database. Queries Q4,



Q5, Q6, and Q8 are answered in this manner. Figure 2 shows the UML diagram of the relational database schema and the queries are defined over this.

Query Q4 is to find all invocations of the *AlignWarp* service that had input parameter '-m 12' as annotation and executed on a *Monday*. The SQL query given below retrieves the invocation IDs for the matching invocations, which is a combination of the entity IDs of the client (invoker) and the service (invokee) that are part of the invocation. The annotations are stored as part of the Notification (Activity) table and the input parameter annotation is part of the *ServiceInvoked* activity. The backend database is MySQL; so date-time functions are specific to that, and in this case, the *Monday* corresponds to the 2nd day of the week.

```
SELECT
  invokee.workflow_id, invokee.service_id, invokee.workflow_node_id,
  invokee.workflow_timestep, invoker.workflow_id, invoker.service_id,
  invoker.workflow_node_id, invoker.workflow_timestep
FROM
  invocation_state_table invocation, entity_table invokee,
  entity_table invoker, notification_table notifications
WHERE
  invokee.entity_id = invocation.invokee_id AND
  invoker.entity_id = invocation.invoker_id AND
  notifications.source_id = invocation.invokee_id AND
  notifications.notification_type = 'ServiceInvoked' AND
  invokee.service_id =
    'urn:qname:http://.../karma/challenge06:AlignWarpService' AND
  notifications.notification_xml LIKE
    '%<ModelMenuNumber>12</ModelMenuNumber>%' AND
  DAYOFWEEK(invocation.request_receive_time) = 2;
```

Once the list of invocation IDs are available for the matching invocations, the *getProcessProvenance* provenance service API method can be called with these invocation IDs as parameters to return the process provenance for each of the invocations.

Queries Out of Scope. Query Q9 is one that is purely based on annotations and the Karma service does not support such complex queries on annotations. While it is possible to perform incremental SQL queries combined with service API calls to answer this query, such queries are not scalable and deemed outside the scope of the Karma provenance system. Karma is part of the LEAD Cyberinfrastructure project and, as in other such projects, there are information services such as the myLEAD personal catalog and Resource Catalog [13] that record generic metadata about data products and services. We expect such systems to answer the bulk of the annotation queries and the provenance is just one piece in the information integration landscape of the virtual organization.

4. DISCUSSION AND CONCLUSION

Going by the categorization of the provenance systems participating in the challenge [10], the Karma framework uses an event driven approach to establish the causal relationship of provenance, and in addition uses the data derivation information encoded in the activities to build the provenance graph (C2.2). It does not require the workflow to be provided (C2.1). This allows Karma to handle *adaptive workflows* seen in weather forecasting, whose definition changes as the workflow is running [8]. Karma



also records both *client and service views* of provenance similar to OPA [9]. Relational databases or XML/RDF seem a popular choice for representing provenance and we leverage the advantages of both (C1.3). While direct SQL queries are used to answer some Challenge queries, this is not the expected mode of usage and the Karma service API is preferred (C1.4). We use the abstraction of *nested workflows* similar to VDL [2] for grouping provenance metadata, allowing control of the depth of the provenance to retrieve (C2.7). We also use the concept of *immediate and deep provenance* to configure the provenance depth by time, similar to ZOOM [3]. While annotations queries are not in the scope of Karma, we were able to address all but one of the Challenge queries (C2.3). Additional queries that can form part of future Challenges include those that make use of the nested workflow abstraction, queries over workflows containing loops, queries over dynamic, adaptive workflows, and queries comparing client and service views of workflows.

We see opportunities to optimize the Karma query interface, especially to tackle some deep provenance queries that currently require client post-processing. These can be easily moved into the service and exposed through the service API, substituting multiple web service calls by the client with more efficient SQL queries by the service. This goes for most of the queries that require client post-processing; the main trade-off is losing the present simplicity of the service API. While annotations are important to get the complete context to provenance, as shown by the several interesting annotation queries in the Challenge, having them as part of the provenance service blurs the line between generic metadata catalogs and provenance services. This behooves the need for meta information services that can integrate provenance and external metadata such as annotations and provide uniform query capability over both.

REFERENCES

1. Bose R, Frew J. Lineage retrieval for scientific data processing: a survey. *ACM Computing Surveys* 2005; **37**(1)1–28.
2. Clifford B, Foster I, Hategan M, Stef-Praun T, Wilde M, Zhao Y. Tracking provenance in a virtual data grid. *Concurrency and Computation: Practice and Experience*, 2007.
3. Cohen-Boulakia S, Cohen S, Davidson S. Addressing the provenance challenge using ZOOM. *Concurrency and Computation: Practice and Experience*, 2007.
4. Czajkowski K, Kesselman C, Fitzgerald S, Foster I. Grid information services for distributed resource sharing. In *HPDC*, 2001.
5. Droegemeier KK, et al. Service-oriented environments for dynamically interacting with mesoscale weather. *Computing in Science and Engineering* 2005; **7**(6)12–29.
6. First Provenance Challenge, Karma results page. <http://twiki.ipaw.info/bin/view/Challenge/Karma> [28 November 2006].
7. Gannon D, Plale B, et al. Service oriented architectures for science gateways on grid systems. In *ICSOC*, 2005.
8. Gannon D, Plale P, Marru S, Kandaswamy G, Simmhan YL, Shirasuna S. Dynamic, adaptive workflows for mesoscale meteorology. In *Workflows for eScience: Scientific Workflows for Grids*, Gannon D, et al. (eds.). Springer, 2006.
9. Miles S, Groth G, Munroe S, Jiang S, Assandri T, Moreau L. Extracting causal graphs from an open provenance data model. *Concurrency and Computation: Practice and Experience* 2007.
10. Moreau L, Ludascher B, et al. The first provenance challenge. *Concurrency and Computation: Practice and Experience* 2007.
11. Simmhan YL, Plale P, Gannon G. A Survey of Data Provenance in e-Science. *SIGMOD Record* 2005; **34**(3)31–36.
12. Simmhan YL, Plale P, Gannon G. A Framework for Collecting Provenance in Data-Centric Scientific Workflows. In *ICWS*, 2006.
13. Simmhan YL, Pallickara SL, Vijayakumar NN, Plale P. Data management in dynamic environment-driven computational science. In *IFIP WoCo9*, 2006.
14. Singh G, et al. A metadata catalog service for data intensive applications. In *Supercomputing*, 2003.
15. Slominski A. Adapting BPEL to Scientific Workflows. In *Workflows for e-science*, Taylor I, et al. (eds.). Springer, 2006.