A Provenance-Integration Framework for Distributed Workflows in Grid Environments

Jing Zhao[†], Fan Sun[†], Carlo Torniai[§], Amol Bakshi[§], Viktor Prasanna[§]
[†]Dept. of Computer Science, [§]Ming Hsieh Dept. of Electrical Engineering
University of Southern California, Los Angeles, CA-90089
{zhaoj, fansun, torniai, amol, prasanna}@usc.edu

Abstract

Provenance information about complex and distributed workflows is a key issue for data quality control and data reliability maintenance in reservoir management. Distributed and integrated environments where different workflows consume and transform data require a comprehensive provenance view. In this scenario provenance collection and integration presents significant challenges. In this paper, we categorize the provenance information into two kinds: the internal provenance which is provenance within workflow instances, and the external provenance which is provenance across workflows. We propose a provenance-integration framework for grid environments which can collect both internal and external provenance, and compose provenance collected from distributed workflows together to get an integrated view. Existing diverse internal provenance models and their corresponding provenance repositories are wrapped as web-services and integrated into our framework in a service-oriented architecture. A provenance index service containing published external provenance is used in the framework to connect internal provenance of multiple workflows by mapping their input/output data objects. The provenance index can also locate provenance requests of users to corresponding provenance services. A set of semantic models are defined in the provenance index service to express the external provenance and the provenance index.

1 Introduction

Provenance is an important issue for complex workflows where various data objects are involved and workflow processing is distributed, for instance, across a grid. Users that are in charge of workflow control, execution, management and analysis need to be able to trace these data objects, so as to know how, when, where, and by whom these data objects were created. These information can help domain experts to understand how trustworthy the data objects are, so that they can achieve good data quality control and data reliability maintenance. Data provenance, which is also referred to as data lineage, can provide domain experts with this kind of information. When used for the data quality control, the provenance information should include ancestral data objects of the data, settings and intermediate results of workflows which create the data, etc. By leveraging data provenance domain experts can go back through the whole derivation history of data objects, debug workflow executions in order to find origin of errors, and learn data accuracy propagations within/between workflows.

In this paper we track data provenance for reservoir management workflows. In reservoir management workflows various data objects are involved, including uncertain reservoir models, reservoir measurements, simulation results, and production schedules. The reservoir management workflows are often distributed in a grid environment. For reservoir management workflows executed in such a distributed environment as grid, having data provenance available is really significant since it results in a more understandable environment for data quality control. But before we discuss the technical details of our work, we first introduce some key characteristics of distributed reservoir management workflows in grid environments as our motivations [20]:

- Diverse and distributed user roles: Typical complex reservoir management workflows involve multiple classes of users and stakeholders, with different specializations and roles across departmental or organizational boundaries [22].

- Integrated environment: Integration in reservoir management involves both application integration and data integration. In a typical reservoir management workflow, domain experts use data objects created by other applications or workflows as input, and integrate a set

of applications (which may have been wrapped as web-services) to produce output data objects through some predefined steps. The output data may be used by other applications or workflows as input. A set of such workflows may be distributed across the whole grid, and may be integrated together to form a higher level workflow. A higher level workflow usually involves work of multiple departments and may last for a long time (e.g. a year).

In order to compose provenance from individual workflows to get an integrated provenance view that combines together multiple workflows, both provenance within a workflow and across workflows have to be collected. In this paper we use *Internal Provenance* to refer to the provenance information within a workflow instance, which includes data derivation relationships, workflow settings, information of underlying systems of the workflow, intermediate results etc. We use *External Provenance* to indicate the provenance across workflows. The external provenance represents which workflows the input data objects come from. It focuses on the input and output of workflows, and does not contain information about internal data of workflows. In this paper we discuss how to capture both internal and external provenance, and we present a solution for *Provenance Integration*, i.e. the procedure of composing internal provenance into external provenance in order to be able to get a comprehensive provenance view.

The characteristics of reservoir management workflows lead to some challenges for provenance integration. The first challenge is the heterogeneity of the provenance models. A provenance model states what provenance information should be collected, and converts unstructured raw provenance data into structured provenance information. A provenance model represents the view of domain experts about the workflow and its data objects. It is built based on the domain and the use of the provenance. Because distributed reservoir management workflows may be created and used by different domain experts with different purposes, diverse provenance models may have been developed for the internal provenance of these workflows. These provenance models may also lead to different provenance storing mechanisms. Considering the large number of existing provenance models and the continuous development of new models, it is not efficient and scalable to unify these internal provenance models. When domain experts retrieve internal provenance of multiple workflows, some methods must be provided to map data objects under different provenance models. The second challenge is how to locate provenance during the integration process. For example, suppose the data object A is used by the workflow W_1 as input. Data object A may be created by another workflow W_2, thus we need to connect provenance captured in different workflows together. Because A may be imported manually (e.g. through copy & paste), users may not know which workflow created it. And since the provenance data is stored in a distributed way in a grid environment, users may also have difficulty to locate A's provenance.

In this paper, we present a provenance-integration framework enabling provenance collection and integration. The main contributions of our work are the following:

1. We present a novel framework for supporting diverse provenance models and easing provenance integration for reservoir management workflows in grid environments. Each provenance repository and its corresponding provenance model is wrapped as a provenance service in the grid, and Service Oriented Architecture (SOA) is used to aggregate provenance services. In this way new provenance service involving new provenance model can be easily integrated into our framework.

2. We develop a provenance index service in the framework, which can connect within a grid the internal provenance of multiple workflows by mapping their input/output data objects, and locate users' provenance requests to corresponding provenance services. External provenance information is published into the provenance index, and is used to connect distributed internal provenance together. A set of semantic models are defined by using ontologies in the provenance index to express the external provenance and the provenance index.

The rest of the paper is structured as follows. Section 2 outlines a simple scenario for provenance integration in reservoir management, which is the motivating example for the discussion in this paper. We present the architecture of our framework in section 3. In section 4 we introduce the semantic provenance models, including the models' definitions and usage. Section 5 describes the implementation of our prototype and deployment framework. In section 6 we survey existing methodologies and tools for data provenance collection and also compare them with our work. We conclude in Section 7.

2 Motivating Example

In this section, we examine an example to motivate our proposed framework. We introduce first the following terminology from oilfield domain [20]. A *well* is an entity that produces oil, water, and gas. A *block* is a set of wells. The production of a block is the sum of the production of its constituent wells. The oil, water, or gas production for a well or a block is often represented by a "recovery curve" or a "decline curve" for that well or block. The production should be under the constraints of *surface facility capacity*, which

refers to the facility and export system capacities over the life of the reservoir.

For the sake of clarity we will consider as motivating example a generic workflow used to forecast and optimize future oil production [8]. In general, the forecasting workflow has five input data sets: block history data, well production data, block data, recovery curve data, and surface facility constraints data. The workflow combines these data sets together to forecast and optimize the future production. The forecasting and optimization program generates large amounts of intermediate results, such as block and well status in each time step during the forecasting procedure. Domain experts need to access these intermediate data so that they are able to inspect the workflow and check data's quality and trustworthy. Therefore we store these intermediate results and their derivation relationships as a significant part of the internal provenance of our workflow.

To prepare the inputs for the forecasting workflow requires complex processes. The procedure generating *well production data* and *block data* involves several workflows which employ lab tests, seismic simulators, and production simulators etc. In the meanwhile, complete surface facility constraints data should consider factors like fluid properties, surface equipment, and even market and transportation conditions. Data will flow across workflows of different departments. In this forecasting workflow, the external provenance records which other workflow instances create the data objects contained in *well production data*, *block data*, and *surface facility constraints data*.

Since workflows may be processed by multiple departments and focus on different domains (e.g. reservoir engineering, production engineering and market management), they usually have their individual internal provenance models and repositories. Without a unified provenance model, one data object may have different names and may be in different forms in different workflows. The heterogeneity and distribution of the internal provenance becomes a major hurdle for domain experts from various departments to track data provenance across multiple workflows.

3 Architecture

Our provenance-integration framework employs Service Oriented Architecture (SOA) in a grid to integrate distributed provenance repositories. Distributed workflows and their heterogeneous internal provenance models lead to distributed provenance repositories. Every provenance repository is wrapped as a provenance service in the grid, which provides detailed internal provenance information when given provenance query requests. Users submit provenance query requests to ask for the internal provenance about some data objects. The data objects are usually outputs of workflows whose provenance is stored in the provenance repository.

There are two problems to be addressed when dealing with the distributed provenance services. As we have discussed above, when domain experts track provenance of a data object across workflows (thus across provenance repositories), the heterogeneity of internal provenance models may cause the same data object to have different names and data formats in different provenance repositories. Thus when domain experts want to retrieve internal provenance information from some provenance service, they need to know the name of the data object (and some other metadata) in that provenance service. Besides, domain experts must be able to locate that provenance service. Sometimes the distribution environment of the grid may prevent them from knowing the addresses (the URL of the provenance services) of all the provenance services.

To solve these two problems, we design and implement a provenance index service which can be seen as a "light" central node connecting distributed provenance services together. By using this index service we can integrate distributed provenance services and heterogeneous provenance models into one framework. The provenance index service can map the input data objects of a workflow instance to the output data objects of other workflow instances. Moreover, it records the URLs of provenance services so that domain experts can use it to locate any provenance service.

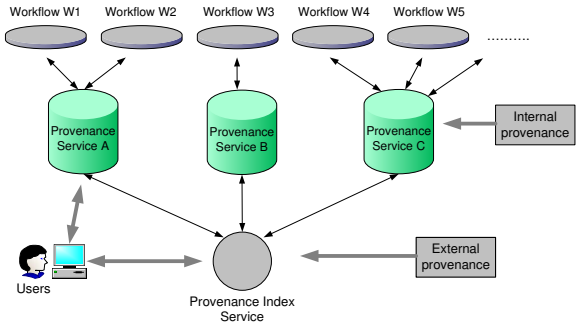

Figure 1. Architecture of the provenance-integration framework.

Fig. 1 depicts the architecture of the provenance-integration framework. There are three levels in our framework. In this first level, diverse workflows are processed and provenance information is collected. The provenance information is stored in provenance repositories as internal provenance. The provenance repositories have been wrapped as provenance services, and these provenance services can be seen as the second level of the framework. One provenance repository may store provenance data of several workflows. Each time domain experts run a workflow, the provenance of this *workflow instance* is recorded.

Different workflow instances may be of the same workflow type but have different workflow settings and input data objects. Since many workflows are run periodically, a large amount of provenance data records are stored in most provenance repositories. Each provenance service provides a set of interfaces for retrieving its internal provenance. It is registered with its interfaces at the provenance index service, and publishes external provenance information to the provenance index service. By using the provenance index service, which is the third level of our framework, we can learn which provenance service has stored the detailed internal provenance for a specific data object, as well as the address of the provenance service. We leverage this provenance index service to integrate within the framework internal provenance collected from distributed workflows with diverse provenance models.

The proposed framework can provide enough scalability to support provenance tracking across distributed workflows for the following reasons:

- Each provenance service can be built independently of the workflow, thus we do not need to adjust the original workflow. A new workflow with the same internal provenance model can easily use the existing provenance service in the grid to publish its provenance information.
- The distributed architecture in the grid environment avoids performance bottleneck brought by traditional centralized architecture [25, 3]. In our framework, the large amounts of the internal provenance data records are stored in distributed provenance repositories across the grid. The provenance index service only records some meta-data and the external provenance for each provenance service, which generates "light" workload.
- New provenance services can be easily built in the grid so as to be added into our provenance-integration framework. After a new provenance service is built, the service administrator will register the new service to the provenance index service with some required information, which includes the address of the service, the involved workflows, types of input and output data objects of the workflows (i.e. the external provenance schema), and the querying interfaces of the service etc.

In our architecture, the index server acts as a central server thus appears to be a single point of failure. We plan to employ a replication strategy to improve the reliability and availability of the central index server. We will arrange several machines as the replication servers of the index server. Every time a provenance service sends external provenance to the index server, the index server will make several copies of the data and store them in the replication servers. When the workload of the index server is not heavy, the replication work is done immediately when the external provenance data is received. But when the index server is busy, it can delay the replication work. If the index server is out of service temporarily or permanently, one of the replication server will be selected as the new index server. In the meanwhile, the external provenance of each provenance service will be collected again to form a fresh and complete external provenance data set.

4 Provenance Index Service

In this section we describe the details of the provenance index service. First we will present its models which are used to represent the external provenance, match data objects from different workflows, and describe the interfaces provided by different provenance services. Then we will show the usage of the provenance index service.

4.1 Modeling

We define three models in the provenance index:

- Data Model. This model is defined as a schema for the external provenance. Data objects and workflow instances are defined in this model. We also use this model to match data objects from different workflows.
- Domain Model. It is a semantic model which expresses explicitly the domain knowledge contained in provenance information. Domain entities and their relationships are defined in this model, and are mapped to data objects captured in provenance information.
- Provenance Service Model. This model captures the semantics of provenance services so that they can be located and invoked. This model is imported into the data model so as to express the relationships between workflows and provenance services.

We use ontologies to define these models. We choose ontologies because they are good at expressing the classification of data objects and domain concepts, and their properties and relationships. Fig. 2 depicts the *Data Model*'s ontology schema. The model captures the input/output relationships between data objects and workflow instances. It also records who run the workflow and create the data objects. It is important to record this information because sometimes the creator of a data object can reveal its quality. Besides, when domain experts cannot understand the internal provenance captured in a workflow, they can find answers from the person who run that workflow. Since the main use of our provenance is for data quality control, we also define the data quality in this model for future extension.

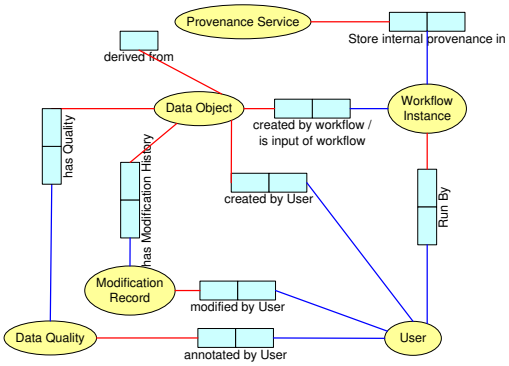

Figure 2. Ontology schema of the data model.

To match the same data object appearing in different workflow instances, we define a 5-tuple $D =< N_e, N_i, W, S, T >$ in the *Data Model* where

- N_e is the data object's name/ID defined in external provenance (i.e. the "public" name in the provenance index),
- N_i is the data object's name/ID defined in some internal provenance (which can be seen as a "private" name of the data object),
- W is the workflow instance in which the data object is involved,
- S is the provenance service which stores $W's$ internal provenance,
- T indicates the relationship between the data object and the workflow instance. E.g. T can specify whether the data object is the input or output of the workflow instance.

In this tuple N_e and N_i may be not the same since a data object may have a different name/ID in some workflow's internal provenance. But since one data object has only one "public" name N_e in the provenance index, we can link multiple tuples with the same N_e so as to match the same data objects appearing in different workflows. Fig. 3 depicts the logic view of the tuple. With a "private" name/ID of a data object (e.g. the shading oval in Fig. 3), the tuple model can easily find the corresponding provenance service (the shading rectangle in Fig. 3) which stores the data object's detailed internal provenance. Thus in some meaning this tuple can be seen as an "index" structure.

The *Domain Model* is defined by an ontology that models the domain entities. Data objects contained in external provenance will all be mapped to this domain ontology. With the help of this domain model domain experts can understand the data objects from the domain level. In the model we use a 4-tuple $D_o =< K, N_d, C, P >$ to define a domain object where K is the class of the domain object, N_d is the name of the domain object, C is a list of domain objects $D_{o1}, D_{o2}, D_{o3}, ...$ which are contained by D_o, and P is the domain object which contains D_o. In this tuple we use C and P to indicate the children-parent relationships among domain entities. We also define other relationships among domain objects. The tuple $R =< D_d, D_r >$ is used to express these relationships where D_d and D_r are the two domain objects between which the relationship is acting. A detailed description of the model can be found in [24].

The *Provenance Service Model* consists of the addresses of the services and the interfaces of the services. Users invoke interfaces to retrieve internal provenance from a provenance service located in a certain address. An interface provides a list of parameters for users to specify the internal provenance they want to get. For different data objects the same provenance service may provide different interfaces.

We use a 3-tuple $S =< N_s, A_s, I >$ to present the provenance service model where the N_s is the name of the provenance service, the A_s is the address of the service, and $I = I_1, I_2, I_3, ...$ defines a list of interfaces. For each interface I_i, we define a 2-tuple $I_i =< D, P >$ where

- $D = D_1, D_2, D_3, ...$ is the list of data objects whose provenance can be retrieved by using the interface I_i,
- $P = P_1, P_2, P_3, ...$, where each P_i is a 2-tuple $< N_i, V_i >$ represents the parameters of the interface. Here N_i is the name of the parameter P_i, and V_i is the value range of P_i.

4.2 Usage

Each provenance service is registered in the provenance index service. When a workflow instance is processed, the internal provenance is stored in the provenance repository of some provenance service. In the meanwhile, some basic information of the workflow instance (e.g. the user who run the workflow) and the input and output data objects are sent to the provenance index. These external provenance information are modeled using the ontologies defined in section 4.1.

When using our provenance-integration framework, a user can start checking internal provenance from a provenance service first. Assume the user checks the internal provenance through the provenance service *A* first, and he/she finds that one important data object is imported from another workflow and its provenance information is not stored in *A*'s repository. The provenance service *A* will contact the provenance index service, inquiring which workflow created the data object, and from which provenance service the provenance information can be retrieved. The

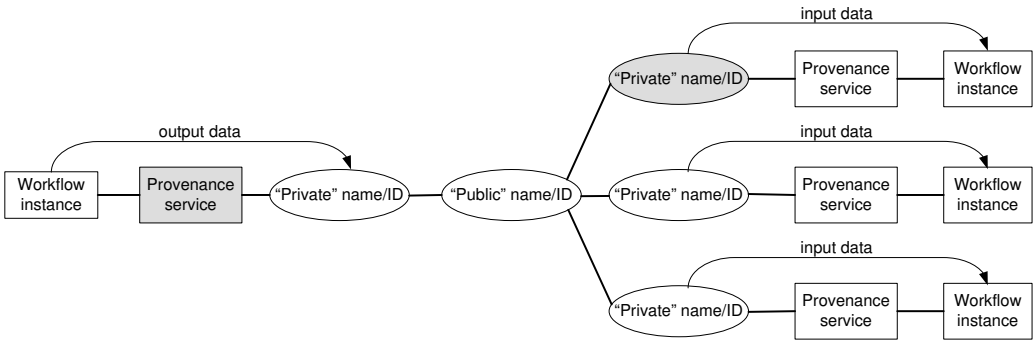

Figure 3. The logic view of the tuple in the data model.

provenance index service will map the received data object's name/ID to the "public" name/ID stored in its external provenance, and use the "public" name/ID to search the local database for the corresponding provenance service. The provenance index service will also retrieve the address of the target provenance service (suppose it is provenance service *B*), the data object's name/ID in service *B*, and the interface of service *B* which defines how to invoke *B* to retrieve the required provenance. These information will be replied to service *A* which will send it back to the user. The user can customize the request of internal provenance according to the parameters of the provenance service *B*'s interface. Then the user will submit a query request to service *B* for the required internal provenance. This process may repeat until the user has got enough provenance information.

Users can also interact with the provenance index service directly to browse the external provenance. The unified external provenance data model and the domain model can help users to have a higher-level overview. In this higher-level overview, different workflow instances are connected with the external provenance, and users can track the data flow across multiple workflow instances. The external provenance only deals with workflow instances and their input/output data objects, but when users want to inspect some workflow instance and check the detailed internal provenance, the provenance index service will find the corresponding provenance service and the interaction between users and the provenance service is the same with the one described above.

5 Implementation

In this section, we describe our implementation of the provenance-integration framework. We start from an internal provenance repository and its provenance service. We still use the forecasting workflow as an example. Considering the large amount of the data records, we stored the internal provenance of the forecasting workflow in a separate relational database. We utilized the Oracle database [31] to develop our database. The relational database tables can be seen as the internal provenance model of the forecasting workflow. We used Apache Axis2 [28] to wrap the provenance repository of the forecasting workflow as a web-service. The provenance service provides a set of interfaces for retrieving the internal provenance information. By giving different parameter values users can specify the level and granularity of the provenance information. For example, users can check the intermediate results for a block or just a well during as short as one-individual-month time or as long as several years time.

Every provenance service registers itself to the provenance index service, and they can obtain the semantic *Domain Model* defined in the provenance index. Therefore in our implementation the *Domain Model* is distributed as a "public" model. After processing one workflow instance, a provenance service annotates the domain concepts defined in the *Domain Model* to the workflow's input/output data objects, and publishes these data objects and the general information of the workflow instance to the provenance index service as external provenance. In the provenance index service, we used Web Ontology Language (OWL) to define the semantic models, and used Jena [29] as the developing framework. For the *Provenance Service Model* we used the OWL-S [15] ontology to represent web-service descriptions. Data records are written as ontology items and are stored in an ontology database.

After one execution of the forecasting workflow which forecasts the oil production in 10 years, about 24,000 internal provenance tuples were generated and stored in the Oracle database. In the meanwhile, only around 250 tuples were generated as the external provenance. In the forecasting workflow, each input data set is created by one or more other workflows. In most of these workflows, a large amount of internal provenance will be collected (which may be even more than the internal provenance collected in the forecasting workflow). If we store all the provenance information, including all the internal provenance collected

from different workflows, in a central database, the database will store huge amounts of data. Querying provenance information in such a huge database will be very slow. Our provenance-integration framework distributes the internal provenance data in different provenance services across the grid, thus avoids storing large amounts of data in a central database. And because the size of external provenance data for each workflow instance is not large, our index service will not become the bottleneck of the framework.

The provenance index service acts as a bridge between different provenance services when domain experts track provenance across workflows. When domain experts want to get the internal provenance about some data object, the provenance index service will tell them where to retrieve the provenance (i.e., which provenance service to invoke) and how to retrieve information (i.e., the interfaces to invoke the provenance service). Users invoke provenance services to retrieve provenance information according to these metadata. Thus in our implementation, rather than transferring the provenance data, the provenance index service deals only with the information required to access provenance services. This implementation can decrease the workload of the provenance index service.

6 Related Work

Provenance has been widely acknowledged and discussed in the e-Science field. A lot of work has been done covering different aspects of provenance, including provenance collection, modeling, representation, storing, and application. Simmhan et al. has listed some of them in a survey [19], in which they presented provenance work in several e-Science projects including Chimera [10, 27, 9], myGrid [25, 26], CMCS [17, 16], ESSW [3, 11] and Trio [23]. Most of them have been implemented in a grid environment. In the following we will discuss these existing work and compare them with our work.

In the myGrid project, workflows are defined by using XScufl language and are executed on the Taverna workflow engine [12] which is a workflow engine developed for grid environments. The provenance information is automatically logged during the workflow execution. MyGrid's provenance model includes four different levels: process level, data level, organization level and knowledge level. Semantic web technologies are employed to link domain knowledge with the provenance information, thus the raw data provenance log is converted into domain-knowledge-enriched provenance. The provenance captured in myGrid can be used for data quality verification.

Chimera uses provenance to track the derivation path of a data product so that scientists can reproduce a data product and verify an experiment. Chimera uses Virtual Data Language (VDL) to define its workflow, and Virtual Data Catalog (VDC) to represent the provenance data. VDC is defined by a query-able Virtual Data Schema, which can also be seen as Chimera's provenance model. VDL can also be used to query the VDC.

In these provenance projects capturing provenance data is tightly coupled with the workflow execution environment [1]. Provenance information can be captured automatically during the workflow execution because of the existence of a workflow engine. However, workflows are often executed in an open distributed environment. The Provenance Aware Service Oriented Architecture (PASOA) project [13] provides an interoperable way to collect provenance in a grid environment. PASOA assumes every workflow component can be wrapped as a web-service, and defines an open protocol among these web-services to capture provenance.

PASOA can be seen as a generic way for provenance recording and can be used in a distributed environment. However, PASOA still focuses on the provenance within a single workflow instance. We cannot use PASOA to integrate provenance across workflow instances or capture integration relationship between workflow instances. VisTrail [18, 6, 7] considers the provenance between workflows. VisTrail collects differences between different versions of a workflow. However, VisTrail's provenance is more focusing on the workflow evolution, thus is still different from our external provenance. And since VisTrail's workflow instances are all based on one basic workflow and only have slight differences in parameters and components, VisTrail does not need to support various workflow models.

The Kepler project [14] provides a workflow framework which can support various types of workflows. Based on the workflow framework, Kepler also develops a provenance framework [1]. This framework can handle different provenance models, and has data, execution, and workflow provenance capabilities. Kepler seems to be a powerful framework for the e-Science field. But in some other fields, especially in some large corporation environments, even if we can deploy workflows in the same grid, since workflows are developed by different departments, it is hard to run all the workflows on a single workflow framework. Under this circumstance we need a more scalable and flexible way to collect provenance under different provenance models, and to compose provenance captured in different workflow instances together to get an integrated provenance view.

Besides e-Science, provenance is also defined and used in many other fields. In database systems Bunman et al. define provenance as "why" provenance, which refers to the source data that had some influence on the existence of the data, and "where" provenance which refers to the location(s) in the source databases from which the data was extracted [4, 5]. In the health care field provenance has been applied in distributed organ transplant management [2]. The provenance data may be stored and recorded in

different hospitals, thus their work aims to be applied in a distributed scenario. However it does not support various provenance models, since different hospitals can share the same provenance model.

In our provenance-integration framework, we employ a central index based service to connect distributed provenance services. This architecture has been used in some other areas. For example, in the WBEM (Web-Based Enterprise Management) [32] architecture, a WBEM server, which acts like our index service, utilizes a previously-created model to determine which real service providers it should communicate with in order to handle clients' requests. Some early peer-to-peer systems, such as Napster [30], also use centralized servers to maintain lists of connected systems and index of the files they provided. We use this architecture in our framework since it fits our internal/external provenance model, and can provide better scalability than the traditional client-server architecture.

7 Conclusion

In this paper we have presented a provenance-integration framework which integrates provenance across reservoir management workflows distributed in a grid environment. We have described the grid-based distributed environment of reservoir management workflows in which different workflows may employ different models to capture provenance and may store provenance in different repositories. This environment brings new challenges to domain experts who need to track provenance across workflows for data quality control. To solve this problem we first defined two different kinds of provenance: the "internal" provenance and "external" provenance. Then we designed a provenance-integration framework in which distributed provenance repositories are wrapped as distributed provenance services, and a provenance index service is used to connect these provenance services. In our framework, internal provenance is stored and retrieved from provenance services while external provenance is recorded in the provenance index service. A set of models are defined in the provenance index service which are used to represent the external provenance, match data objects from different workflows, and describe the interfaces provided by different provenance services. Semantic technologies are used to define these models and express the domain knowledge contained in the provenance data.

8 Acknowledgments

This research was funded by CiSoft (Center for Interactive Smart Oilfield Technologies), a Center of Research Excellence and Academic Training and a joint venture between the University of Southern California and Chevron. We are grateful to the management of CiSoft and Chevron for permission to present this work. We are thankful to Will Da Sie and Laurent Pianelo (Chevron Corporation) for sharing a wealth of domain knowledge, and providing feedback on our prototypes.

References

[1] I. Altintas, O. Barney, and E. Jaeger-Frank. Provenance Collection Support in the Kepler Scientific Workflow System. In First International Provenance and Annotation Workshop, Chicago, USA, May 2006.

[2] S. lvarez, J. Vazquez-Salceda, T. Kifor, L. Z. Varga, and S. Willmott. Applying Provenance in Distributed Organ Transplant Management. In First International Provenance and Annotation Workshop, Chicago, USA, May 2006.

[3] R. Bose and J. Frew. Composing Lineage Metadata with XML for Custom Satellite-Derived Data Products. In Proceedings of the 16th Conference on Scientific and Statistical Database Management, Greece, June 2004.

[4] P. Buneman, S. Khanna, and W.C. Tan. Why and Where: a characterization of data provenance. In Proceedings of International Conference on Database Theory (ICDT 2001), London, UK, January 2001.

[5] P. Buneman, S. Khanna, and W.C. Tan. Data Provenance: Some Basic Issues. In Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science, New Delhi, India, December 2000.

[6] S.P. Callahan, J. Freire, C.E. Scheidegger, C.T. Silva, and H.T. Vo. Towards a Provenance-Enabled ParaView. In Proceedings of the Second International Provenance and Annotation Workshop (IPAW 2008), Salt Lack City, USA, June 2008.

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

[8] Luca Cosentino. Integrated Reservoir Studies. Editions Technip, Paris, 2001, pp.281-286.

[9] I. Foster, J. S. Vöckler, M. Wilde, and Y. Zhao. The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration. In Biennial Conference on Innovative Data Systems Research (CIDR 2003), Asilomar, CA, January 2003.

[10] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation. In Proceedings of the 14th Conference on Scientific and Statistical Database Management, Edinburgh, Scotland, UK, July 2002.

[11] J. Frew and R. Bose. Earth System Science Workbench: A Data Management Infrastructure for Earth Science Products. In Proceedings of the 13th Conference on Scientific and Statistical Database Management, Fairfax, Virginia, USA, July 2001.

[12] M. Greenwood, et al. Provenance of e-Science Experiments - experience from Bioinformatics. In Proceedings of The UK OST e-Science second All Hands Meeting 2003 (AHM 2003).

[13] P. Groth, M. Luck, and L. Moreau. A protocol for recording provenance in service-oriented grids. In Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS 2004), Grenoble, France, December 2004.

[14] B. Ludäscher, et al. Scientific Workflow Management and the Kepler System. Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows, 2005. http://kepler-project.org/

[15] D. Martin, et al. OWL-S: Semantic markup for web services. http://www.ai.sri.com/daml/services/owl-s/1.2/overview/

[16] J. Myers, C. Pancerella, C. Lansing, K. Schuchardt, and B. Didier. Multi-Scale Science, Supporting Emerging Practice with Semantically Derived Provenance. In International Semantic Web Conference (ISWC) Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, Sanibel Island, Florida, USA, 2003.

[17] C. Pancerella, et al. Metadata in the Collaboratory for Multi-Scale Chemical Science. In Dublin Core Conference, 2003.

[18] E. Santos, L. Lins, J.P. Ahrens, J. Freire, and C.T. Silva. A First Study on Clustering Collections of Workflow Graphs. In Proceedings of the Second International Provenance and Annotation Workshop (IPAW 2008), Salt Lack City, USA, June 2008.

[19] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. In SIGMOD Record 34(3): 31-36, 2005.

[20] R. Soma, A. Bakshi, V. K. Prasanna, and Will Da Sie. A Model-Based Framework for Developing and Deploying Data Aggregation Services. In 4th International Conference on Service Oriented Computing, Chicago, USA, December 2006.

[21] F. Sun, V. K. Prasanna, A. Bakshi, and L. Pianelo. Workflow instance detection: Toward a Knowledge Capture Methodology for Smart Oilfields. In Proceeding of the 2008 IEEE International Conference on Information Reuse and Integration (IEEE IRI 2008), Las Vegas, USA, July 2008.

[22] G.C. Thakur, and A. Satter. Integrated Waterflood Asset Management. PennWell Books (1998).

[23] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, California, USA, January 2005.

[24] Cong Zhang, Viktor Prasanna, Abdollah Orangi, Will Da Sie, and Aditya Kwatra. Modeling Methodology for Application Development in Petroleum Industry. In IEEE International Conference on Information Reuse and Integration, Las Vegas, USA, July 2005.

[25] J. Zhao, C. A. Goble, R. Stevens, and S. Bechhofer. Semantically Linking and Browsing Provenance Logs for E-science. In International Conference on Semantics of a Networked World (ICSNW 2004), Paris, France, June 2004.

[26] J. Zhao, C. Wroe, C. A. Goble, R. Stevens, D. Quan, and M. Greenwood. Using Semantic Web Technologies for Representing E-science Provenance. In International Semantic Web Conference, Hiroshima, Japan, November 2004.

[27] Y. Zhao, M. Wilde, and I. Foster. Applying the Virtual Data Provenance Model. In First International Provenance and Annotation Workshop, Chicago, USA, May 2006.

[28] Apache Axis2: http://ws.apache.org/axis2

[29] Jena - A Semantic Web Framework for Java. http://jena.sourceforge.net

[30] Napster: http://www.napster.com

[31] Oracle database: http://www.oracle.com/database

[32] Web-Based Enterprise Management: http://en.wikipedia.org/wiki/Web-Based_Enterprise_Management

[33] Windows Workflow Foundation: http://msdn.microsoft.com/en-us/netframework/aa663328.aspx