Provenance for Scientific Workflows Towards Reproducible Research

Roger Barga[1], Yogesh Simmhan[1], Eran Chinthaka Withana[2], Satya Sahoo[3], Jared Jackson[1], Nelson Araujo[1]
[1]Microsoft Research, Redmond WA 98052
[2]Indiana University, Bloomington IN 47405
[3]Wright State University, Dayton OH 45435

1 Introduction

eScience has established itself as a key pillar in scientific discovery, continuing the evolution of the scientific discovery process from theoretical to empirical to computational science [13]. Extensive deployment of instruments and sensors that observe the physical and biological world are bringing in large and diverse data to the reach of scientists. Often, that data is more frequently shared due to the cost of the instrumentation or because of the desire to address larger scale and/or cross-discipline science such as climate change. There is a tangible push towards building large, global-scale instruments [2][3] and wide deployment of sensors [1] with the data they generate being shared by a large collaboration which has access to the data generated by these instruments. Indeed, funding agencies and publishers are starting to insist that scientists share both results and raw datasets, along with the provenance for how the result was produced from the raw dataset(s), to foster open science [4].

Scientific workflows have emerged as the de facto model for researchers to process, transform and analyze scientific data. These workflows may run on the users desktop or in the Cloud and the workflow framework is geared towards easy composition of scientific experiments, allocation and scheduling of resources, orchestration and monitoring of execution, and collecting provenance [20]. The goal of the Trident Scientific Workflow System is to provide a specialized programming environment to simplify the programming effort required by scientists to orchestrate a computational science experiment.

1.1 Trident Scientific Workflow System

In designing Trident [9][5], our goal was to leverage the existing functionality of a commercial workflow management system to the extent possible and focus development efforts only on functionality required to support scientific workflows. The result is a much smaller code base to maintain, improving sustainability, and a thorough understanding of requirements unique to scientific workflows. Trident is implemented on top of Windows Workflow Foundation [6], a workflow enactment engine included in the Windows operating system. Windows Workflow Foundation is highly extensible. All activities in Windows WF derive from a common .NET base class and an extensible development model enables the creation of domain specific activities that can then be used to compose workflows useful and understandable to domain scientists. In addition users can add new infrastructure services to the workflow runtime, such as automatic provenance capture.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

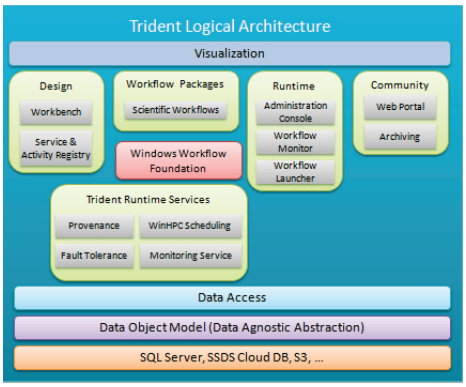

Figure 1: Diagram of the Trident logical architecture

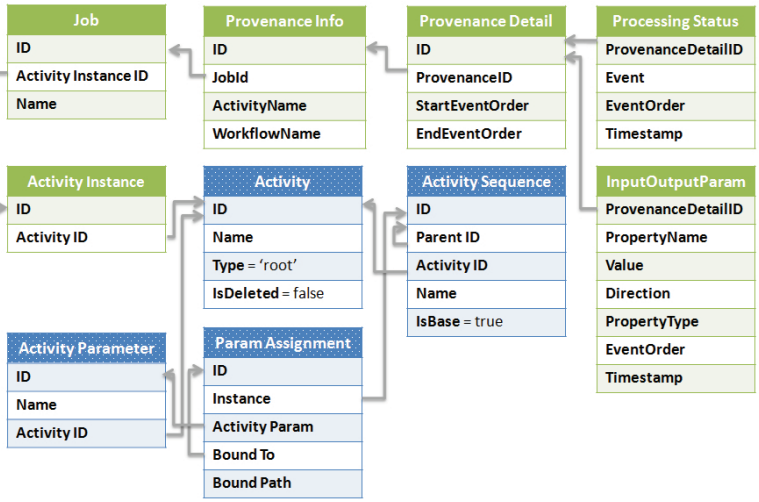

Figure 2: Provenance Data Model. Entities in dotted blue describe the workflow schema that is populated during composition. Entities in solid green describe the provenance schema that is populated at runtime.

The key elements of the Trident architecture, illustrated in Figure 1, include a composer that enables users to visually author a workflow using a library of existing activities and complete workflows. The registry serves as a metadata catalog of known datasets, services, workflows, activities, and computational resources, and it maintains state for workflows that are currently active. Administrative tools allow users to register and manage compute resources, track workflows currently running, along with a workflow launcher that supports scheduling of workflows. Community tools include a web service to launch workflows from any web browser, access to a repository of workflow results, and a facility to publish and share workflows with other scientists. At the lowest level of Trident is a data access layer that abstracts the actual storage service used from the workflows, so a workflow can read and/or write data objects that are transparently mapped to the target data source. We currently support a default XML store and SQL Server for local storage, along with Amazon S3 and SQL Azure for cloud storage.

The rest of the paper is organized as follows: Section 2 describes the Trident provenance data model, along with the capture, storage and querying of provenance, while Section 2.4 describes the workflow evolution framework in Trident that tracks the evolution of workflow definition and the results produced by individual workflow version, related work is presented in Section 4, and we summarize our paper in Section 5.

2 Provenance in Trident

The Trident Workbench provides an integrated way to collect, store, query, and view provenance for scientific workflows [18]. Windows WF generates low level events during every step of workflow execution, from the point a workflow is loaded for execution to workflow termination. Trident intercepts and routes these event through a publish-subscribe API; C# libraries we provide allow user defined provenance events to be published. Trident enhances basic Windows WF tracking by adding automated logging of workflow input and output parameters, execution events and data flows, including external applications launched from the workflow and system calls. In addition, Trident tracks workflow evolution provenance [14], to record changes to the individual workflows over time and correlates versioned workflows with the results they generate [8]. The provenance data collected by Trident is compatible with the emerging Open Provenance Model standard [17] and different

data stores are supported for storing provenance. In the following sections, we discuss the interoperable data model used to represent provenance in Trident, means for distributed collection of provenance from the various workbench components, and its storage and dissemination.

2.1 Provenance Data Model

Provenance information is available as a combination of static, composition information about the workflow - *the workflow schema*, along with dynamic, runtime information about the actual execution of an instance of a workflow - the *provenance schema*. The workflow composition information provides structural knowledge of the workflow, its activities, and their data and parameter type signatures, and is static for a composed workflow version. These are described using the *Activity, Activity Sequence, Activity Parameter* and *Parameter Assignment* entities in the data model shown in Figure 2. The composed workflow is an abstract workflow that can be instantiated with initial parameter and data values, and executed by the workflow engine. Information about the actual workflow execution forms the runtime provenance, and includes the job that represents a local or remote submission of the workflow instance to the workflow engine, the workflow and activity execution times, the parameters to/from each activity instance, and the status of their execution. The *Job, Activity Instance, Provenance Info, Provenance Detail, Processing Status* and *InputOutputParam* entities in Figure 2 record these. Using this two level model of abstract workflow and workflow instance details intuitively corresponds to composition and execution phases of a workflow. It also allows common workflow features that do not change across runs to be stored once while dynamic information unique to each workflow run is accumulated. Having explicit relations between the abstract and instance metadata allows rich queries that combine both workflow structural features and runtime knowledge.

This native data model used in Trident is also compatible with the Open Provenance Model (OPM) specification [17] that evolved at the same time as Trident. OPM allows interoperability with other provenance and workflow systems to allow provenance tracking of data that are reused across workflow systems, as is common in collaborative research projects. Both OPM and Trident use similar entities to represent workflows and activities (OPM Processes), data products and parameters (OPM Artifacts), and relationships like Used and Produced for data dependencies. The OPM model is a subset of the native Trident model and we have demonstrated the ability to export Trident provenance to OPM and import it from other systems to perform interoperable provenance queries, as part of the third Provenance Challenge workshop [19].

2.2 Provenance Collection

The primary source for the abstract workflow details is the workflow composer. Workflow activities imported into the workbench are examined and their signatures extracted into the Activity and Activity Parameter entities. When users compose new workflows or import existing ones and save them, the Activity Sequence and Parameter Assignment entities store the data and control flows present in the workflow.

Collecting runtime provenance information about a workflow instance is more challenging given the distributed nature of workflow execution. The different sources of dynamic information include the *execution service* that submits and triggers workflow job execution, the *Windows WF engine* that orchestrates the workflow instance by invoking activities and the *activities* themselves that may be built-in or user defined, illustrated in Figure 3. The execution service tracks the submission of each workflow instance by the workbench or management studio interface, its status (scheduled, executed, or completed) on a local or remote machine. It is the source for *Job* and *Activity Instance* entities of the data model. The Windows WF engine natively generates tracking events of the workflow execution, such as when the workflow starts and finishes, an activity initializes, executes, and completes successfully or fails, or an exception is raised. Trident event handlers to listen for these built-in events and populate the *Provenance Info, Provenance Detail* and *Processing Status* entities that track the workflow's control flow. Lastly, we use instrumentation in the Trident base activity class to generate custom

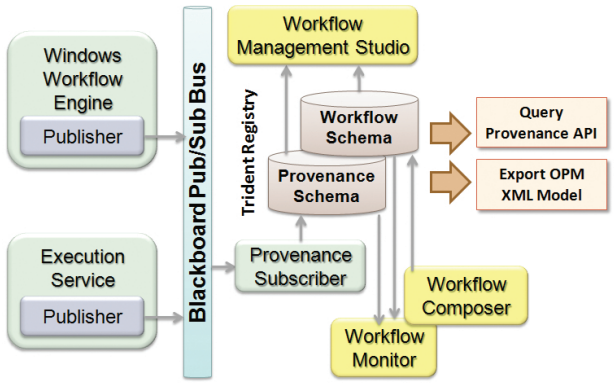

Figure 3: Provenance Architecture

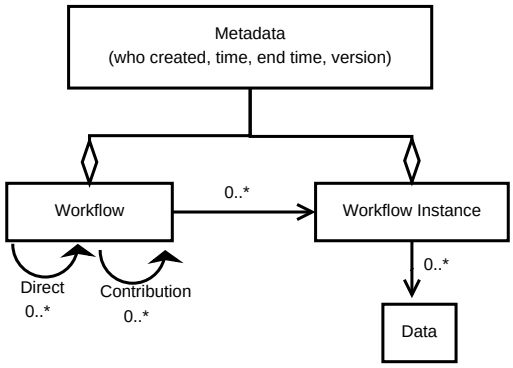

Figure 4: Versioning Data Model within EVF

user events that capture the actual input and output parameter values passed to/ from the activities that provide data flow knowledge used to fill the *InputOutputParam* entity.

Given that these information sources may each be running on a different machine, and remote from where the provenance is stored, we have developed the BlackBoard [21] publish-subscribe, asynchronous messaging framework to distribute provenance events from the sources to the provenance storage. The messages form incremental pieces of the complete provenance information. They are generated by publisher libraries incorporated into the workflow engine and execution Service, and are listened to by a Provenance Service subscriber. Message payloads consist of a list of name-value attributes that identify the workflow instance and activity that the message describes, the timestamp of the message, and details such as the current status of the workflow or activity, input or output parameter values, and exceptions that may have occurred. The provenance service listens to the events and records them in the provenance store in the Trident Registry.

Besides distributed provenance collection, the use of the BlackBoard pub-sub model has two other advantages: (1) it separates the responsibility of message creation by the source from its fault resilient delivery by the BlackBoard broker to the subscriber, and (2) it allows the same set of messages to be delivered to different subscribers with different goals, such as real-time monitoring in a GUI, filtering them to detect workflow failures or anomalies [3], or simply recording them in the provenance store. Both of these features are essential for reliable scaling of workflow instances across distributed resources [21].

2.3 Provenance Storage

The Trident Registry is the central information repository and stores the provenance and workflow metadata. It uses the data access layer to abstract the data model from the storage layer, allowing the use of local and cloud backends. The Registry data model can be easily configured through an XML schema file and supports entities and relationships with different cardinalities.

Trident's complete data model, including the provenance and workflow entities, is created and available in the Registry schema with auto-generated .NET objects for programmatic access all the entities. The provenance service and other Registry users initially create a .NET handle to the active workbench Registry. Inserting a tuple for a particular entity is as simple as instantiating a .NET object of that type and setting its properties. Similarly, relationships between different entity tuples are created from corresponding .NET object references. Depending on the Registry storage backend, storage adapter implementations map the .NET objects to the storage layer, for example, SQL Tables and tuples or Amazon S3 blob objects. This approach makes it easy for .NET services and workbench to easily and intuitively store data in the Registry without programming directly to the storage implementation, and allows backend storage to be switched to suit the needs of the scientific domain.

The Registry also supports queries over the data model using the .NET Language Independent Query (LINQ) mechanism [10]. This provides a collection abstraction for accessing stored items in the data model and allows filter, project, and aggregate operators over the collection. Specifying queries becomes a natural extension of the .NET language and results are returned as collections of .NET objects. LINQ also supports deferred, just in time execution for efficiency. The Registry's backend storage adapters provide the implementation for mapping the LINQ to the access/query model supported by the storage layer, such as, SQL or REST.

2.4 Provenance Query and Dissemination

Provenance is typically used in three ways in Trident: for realtime monitoring of the workflows, for analysis and mining post workflow execution and for exporting to external provenance systems. The workflow monitor GUI allows users to launch and visualize the workflow execution from a user's desktop. Upon workflow launch, the workflow monitor subscribes to provenance messages from the BlackBoard broker for that workflow instance. The light-weight provenance messages published by the workflow engine and execution service allow the workflow monitor to show the current status of each workflow activity as they execute. The monitor also displays resource performance information, such as CPU and memory usage, of the machine running the workflow and activities that are also published as separate events. Additionally, the workflow monitor can pull more detailed information about the workflow execution by querying the provenance service using concise LINQ queries over the Registry data model. This is also possible through the workflow management studio where a user can list of all past and currently executing workflows and retrieve details of the workflow, its execution, the data produced and consumed, and the faults that happened, and step through each activity execution in the workflow to visually replay the execution. This also allows the same workflow to be re-executed for validation. These features provide a powerful tool for scientists to maintain a comprehensive record of their in silico experiments that is useful for verification and reproducibility.

Some of the queries supported by the provenance and workflow data model include:

- *Display the **status statistics** of all **workflows** run **today***: This groups all workflows run today by their status property, and returns and prints the count for workflows for each status.

```
var jobStats =
  // Todays' WFs
  from j in jobs where j.Update.ToShortDate() == DateTime.Now.ToShortDate()
  // Group by status
  group j by j.Status into g
  // WF count for each status
  select new {status = g.Key, count = g.Count()};
// Print stats
foreach(var stats in jobStats) Console.WriteLine(stats.ToString());
```

- *Find the authors who **composed workflows** which have the output of the **SecretTask** activity **passed as input** to the **OnlinePublish***: This queries the data flow of the abstract workflow for those workflows that can potentially publish the output for a proprietary 'SecretTask' activity through an OnlinePublish activity, and identifies the authors of those workflows.

```
var badAuthors = from aseq in
  // Activity OnlinePublish's input is connected to output from activity
  // SecretTask
  (from pa in paramAssigns
   where pa.ActivityParameter.Activity.ActivityClass == "OnlinePublish" &&
         pa.BoundTo.ActivityParameter.Activity.ActivityClass == "SecretTask"
   select pa.ActivityParameter.Activity.ActivitySequences).SelectMany(sq => sq.ToList())
  where aseq.Parent.IsBase select aseq.Parent.Activity.Author; // Authors of parent WF
```

- *Find all **abstract workflows** that have been **executed** with an **activity** that uses the **Needleman-Wunsch** as the genome sequence **aligner** parameter*: This provenance data flow LINQ query selects InputOutput entities

with the 'aligner' activity parameter set to 'Needleman-Wunsch' and traces them back to jobs that ran the workflow instances containing that activity, and returns the abstract workflows for those jobs.

```
var needlemanWorkflows =
  // 1. Activity is a workflow, and ...
  from a in activities where a.IsWorkflow &&
    // 2.A. Job IDs for WF
    (from j in (from ai in a.Instances select ai.Jobs) select j.ID)
  .Intersect
    (from pi in (from io in inOuts where io.Name == "aligner" &&
                                  io.Value == "Needleman-Wunsch" select io.Provenance)
        // 2.B. Job IDs with Needleman-Wunsch activity parameter value
        select pi.JobId
    // Intersection of job IDs from (2.A) and (2.B) has items
    ).Count() > 0   select a;
```

Lastly, provenance for a workflow instance execution stored in the Registry can be published as an XML document following the OPM specification using a prototype tool. This allows workflow provenance to be interoperably shared with the external community for collaboration, for peer review, or as part of a publication that uses that experiment, usable by even users who do not employ Trident as their workflow workbench.

3 Versioning of Workflow Evolution

As researchers use a workflow management system to carry out their computations the workflows evolve as the research evolves and this workflow evolution can be a valuable tool for tracking both the evolution of the research and results created by a specific workflow instance across time. Scientists can trace their research and associated results through time or go back in time to a previous stage and fork a new branch of exploration. And since workflows encapsulate a vast amount of knowledge associated with experiments, tracking the evolution of workflow can help to aggregate this knowledge for later analysis. To support this, Trident provides a workflow evolution framework (EVF) to enables efficient management of knowledge associated with workflow evolution over time. The benefits of the Trident workflow evolution framework include the following:

- *Research evolution*: When a scientist associates their research with a collection of workflow, tracking the evolution of these workflows becomes an approximation to the initial problem of tracking the evolution of their research process. Along with the evolution of a workflow, all the components within it will also evolve from the selection of web services and databases to the implementation of individual activities. Scientists can later examine the result of a workflow execution and reason about how the research evolved to the current state to produce that particular output.
- *Result Comparison*: A given research exploration may evolve in more than one direction. It is important to understand the changes these choices had on the outcome of the research by comparing the difference between the outputs of two or more versions of the same workflow.
- *Attribution*: When a workflow is executed, attribution information such as who performed the experiment, the author of the workflow, the owner of the data products, etc., can be collected at runtime and associated with the final result. This attribution information can be used later to identify issues with data or code quality and to give proper credit to contributors. Also, while carrying out an experiment it is becoming more common to reuse subset graphs within a workflow scientists can utilize not only the algorithms and implementations developed by others, but also the data products generated including optimally derived model parameter configurations. In research, it is not only the technical aspects that matter; sharing and attribution of research can and should be an integral part of research. The Trident workflow management system can access and download subset graphs from sources like myexperiment.org [12] to reduce development costs and track the author with our evolution framework for proper accreditation to the contributors.

3.1 Versioning Model

In order for workflow based research to be reproducible, a versioning strategy needs to consider the workflow, along with the associated data products, parameters, configurations and executable, and bind this information together. The Trident Workflow Evolution framework can support reproducibility by persisting all information about previously executed experiments. If the underlying data management service supports versioned data products, then a scientist using Trident can re-run previously executed experiments.

This versioning model, illustrated in Figure 4, is built on two orthogonal dimensions of workflow evolution, namely direct evolution and contributions. Direct evolution occurs when a user performs one of the following:

- Changes the flow and arrangements of the components within the system;
- Changes the components within the workflow;
- Changes inputs and/or output parameters or configuration parameters to different components within the workflow.

Direct evolution will primarily come from a researchers direct manipulation (editing) of the workflow that is being tracked. On the other hand contributions will track components that are reused from previous system.

One of the unique features of the Trident EVF is that it tracks both direct evolution and contributions to research. Together this contributes towards the existing eco-systems to acknowledge each other's contributions to the existing research and also encourages scientists to share and use existing work. Versioning of workflows and related artifacts is done at three separate stages of execution.

- User explicitly saves the workflow;
- User closes the workflow editor;
- Executing a workflow in the editor: since workflow instances should always be associated with a workflow, EVF requires all the workflows to be saved and versioned before executing them.

This level of granularity does not capture minor edits to a workflow, but in applying EVF in actual use cases we have established a level of sufficiency for this versioning for later retrieval and workflow evolution. Figure 4 also presents the data model used for versioning of objects and the relationships between them. This model is designed such that all the artifacts related to an experiment can be captured and versioned.

3.2 Architectural Features

There are several architectural features that enable the workflow evolution tracking, which are presented below:
Unique Association of Research Artifacts to Workflows As a workflow is executed, the Trident EVF associates the relevant data, parameters, and configuration information, as well as meta-data identifying who performed the experiment (user id), when and where the output was saved in the system, etc. In addition Trident records the lineage information of the workflow. This information is of value in evaluating the end result and can be used to reproduce the research at a later time. Trident records this information in the provenance log using the information model outlined in Figure 4 and associates both provenance log and output result in the Trident Registry.
Automatic Versioning The Trident EVF automatically versions workflows as users edit them. This enables a researcher to later retrieve a previous workflow for viewing or to create new branches from previous points in the workflow evolution. Versioning of the workflow templates inside EVF is comparable to a typical version control system, but EVF also has the ability to work with other versioning systems to support different versions of data products. The Trident EVF provides clearly defined extension points to add new versioning systems. Once a data provider, capable of versioning data products, is registered with the system, EVF will save enough information to retrieve a given version of a data product. When EVF associates a data product with a workflow execution it will include this versioning information so that the correct version of the data product can be retrieved later, in case the scientist is interested in reproducing the research. Also if an extension is registered to handle versioning, EVF will use that extension to automatically retrieve the data and to execute the workflow within Trident. During the

workflow authoring process, the user may make multiple changes to a workflow and possibly save intermediate steps. But in the end only execute the final version of the workflow. Should the scientist opt to delete the previous versions, EVF gives the control to the user to select the versions to persist inside the registry or to remove from the system. This will not only reduce the clutter in the scientist's workspace, but also optimizes the workflow lineage information persistence. However, a user is not allowed to delete a workflow version that is either associated with a result or contributed to a workflow that produced a result.

Navigation through Time Navigating a workflow evolution through time can provide a unique view on the evolution of the research and associated output results. A user may see a change or improvement in the results of tan experiments over time, observe the effects of the different data sets being used, or identify the contributor of a new subset graph (workflow). Trident captures sufficient information to allow a user to select a workflow and navigate previous versions of that workflow through time, visualizing the evolution of both the workflow and associated results. Since the Trident EVF information model associates the workflow instances of a given version of the workflow, scientists can even see the runtime information of each and every workflow execution. We believe that providing this information along a time-line will give users more insight into their research process.

4 Related Work

Provenance in scientific workflows has received attention in the recent past. Several workflow systems support provenance recording, such as Taverna [16] and Kepler [15]. Taverna [16] uses a pure dataflow model for its workflows and its provenance capture is also limited to dataflow provenance. Trident in addition captures the control flow aspects of the workflow and its provenance. Taverna also captures provenance for hierarchical data collections, which Trident does not.

There are also stand alone provenance tracking and storage systems like Karma [11] and PASOA [7]. Karma [11] uses instrumentation of workflow engine and Axis2 web service activities for recording provenance in a database, also using a pub-sub model for event transfer. It has been demonstrated with G-BPEL and ODE workflow engines. Trident uses similar instrumentation but is more tightly integrated with the Windows WF engine and activities. This is partly due to the reuse of existing tracking information provided by Windows WF, and enables features like tracking provenance of interactive user operations on Windows GUI elements like graphs and forms. Karma and PASOA support dual views of provenance tracking, both from the workflow engine and activity/web service. Trident does not make this distinction. However, neither libraries support transparent use of diverse backend storage such as file, database, and cloud that the Trident Registry enables. Despite these differences, there is overlap on the provenance collected by the different provenance systems as demonstrated in the third Provenance Challenge workshop, where provenance data collected in Trident was able to interoperate with the provenance systems of Taverna, Kepler, Karma and PASOA [19].

5 Summary

Scientific workflows have emerged as the de facto model for researchers to process, transform and analyze scientific data. Workflow management systems provide researchers with many valuable and time saving features, from cataloging workflows, workflow activities and web services, to visual authoring and workflow monitoring. But arguably the most valuable service they offer is the automatic capture of provenance data sufficient to establish trust in a result and potentially allow other researchers to reproduce a result. In this paper we have summarized highlights of the workflow provenance that is automatically collected and managed by the Trident Scientific Workflow Workbench. Trident provides an integrated way to collect, store and view provenance for workflows, and supports a range of provenance queries over workflow results. The implementation is based on the ability to intercept and routes low level events through a scalable and high performance pub-sub API.

In addition Trident tracks workflow evolution provenance to record changes to individual workflows over time and correlates versioned workflows with the results they generate. The provenance data collected by Trident is compatible with the emerging Open Provenance Model standard and different data stores are supported for storing provenance.

References

[1] "Fluxnet," http://www.fluxdata.org.

[2] "Large hadron collider (lhc)," http://public.web.cern.ch/public/en/LHC/LHC-en.html.

[3] "Large synoptic sky survey (lsst)," http://www.lsst.org/lsst_home.shtml.

[4] "Nih public access policy," http://publicaccess.nih.gov/policy.htm.

[5] "Project trident download," http://research.microsoft.com/en-us/collaboration/tools/trident.aspx.

[6] "Windows workflow foundation (winwf)," http://en.wikipedia.org/wiki/Windows_Workflow_Foundation.

[7] *Recording and Using Provenance in a Protein Compressibility Experiment*, 2005.

[8] *Versioning for Workflow Evolution*, Chicago, IL, June 2010. [Online]. Available: http://www.cct.lsu.edu/~kosar/didc10/papers/didc06.pdf

[9] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan, "The trident scientific workflow workbench," *eScience, IEEE International Conference on*, vol. 0, pp. 317–318, 2008.

[10] D. Box and A. Hejlsberg, "Linq: .net language-integrated query," http://msdn.microsoft.com/en-us/library/bb308959.aspx.

[11] B. Cao, B. Plale, G. Subramanian, E. Robertson, and Y. Simmhan, "Provenance information model of karma version 3," in *IEEE 2009 Third International Workshop on Scientific Workflows*, 2009.

[12] C. A. Goble and D. C. De Roure, "myexperiment: social networking for workflow-using e-scientists," in *WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science*. New York, NY, USA: ACM, 2007, pp. 1–2.

[13] T. Hey, S. Tansley, and K. Tolle, "The fourth paradigm: data-intensive scientific discovery," 2009, http://research.microsoft.com/en-us/collaboration/fourthparadigm/.

[14] L. Lins, D. Koop, E. Anderson, S. Callahan, E. Santos, C. Scheidegger, J. Freire, and C. Silva, "Examining statistics of workflow evolution provenance: A first study," in *Scientific and Statistical Database Management*. Springer, 2008, pp. 573–579.

[15] T. McPhillips, S. Bowers, D. Zinn, and B. Ludscher, "Scientific workflow design for mere mortals," *Future Generation Computing Systems*, 2009.

[16] P. Missier, S. Sahoo, J. Zhao, C. Goble, and A. Sheth, "Janus: from workflows to semantic provenance and linked open data."

[17] L. Moreau, J. Freire, J. Futrelle, R. McGrath, J. Myers, and P. Paulson, "The open provenance model," *University of Southampton*, 2007.

[18] Y. Simmhan, C. van Ingen, A. Szalay, R. Barga, and J. Heasley, "Building reliable data pipelines for managing community data using scientific workflows," in *e-Science, 2009. e-Science '09. Fifth IEEE International Conference on*, 9-11 2009, pp. 321 –328.

[19] Y. Simmhan and R. Barga, "Analysis of approaches to supporting the open provenance model in the trident workflow workbench," *Future Generation Computing Systems (in Review)*, 2010.

[20] Y. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *ACM SIGMOD Record*, vol. 34, no. 3, p. 36, 2005.

[21] M. Valerio, S. Sahoo, R. Barga, and J. Jackson, "Capturing Workflow Event Data for Monitoring, Performance Analysis, and Management of Scientific Workflows," in *IEEE Fourth International Conference on eScience, 2008. eScience'08*, 2008, pp. 626–633.