

Provenance in High-Energy Physics Workflows

The adoption of large-scale distributed computing presents new opportunities and challenges for the physicists analyzing data from the Large Hadron Collider experiments. With petabytes of data to manage, effective use of provenance is critical to understanding the results.

The Large Hadron Collider (LHC) at CERN will produce data at a new energy scale of great interest to particle physicists. With proton–proton collisions at a center-of-mass energy of 14 TeV, many of the collisions between the protons' constituent partons will be above the energy scale at which the symmetry between weak and electromagnetic interactions can be restored and will appear as the more fundamental electroweak interaction. Two general-purpose experiments, ATLAS and the Compact Muon Solenoid (CMS), will study these high-energy processes in great detail, and the results have the potential to illuminate the processes by which electroweak symmetry is broken and fundamental particles acquire mass—it could even inform us about the makeup of the cold dark matter that dominates the universe's mass. However, researchers must overcome significant computing challenges to provide the individual physicist with the means to effectively and efficiently pursue analysis of such data.

In this article, we examine the role of provenance in several projects in which members of

our group have participated, with an emphasis on using data provenance to improve the reproducibility of the physics results from the final steps of data analysis. Although the problems we discuss are domain specific, the solutions we describe have broader applications.

Physics Workflows

In high-energy physics (HEP) experiments, computing is typically designed around a model that organizes computationally intensive data processing in a centralized manner akin to widget production in a factory. In this model, small teams of experts working in tightly controlled environments reconstruct and reduce data sets to subsamples containing high-level physics objects. The resulting organized production tasks are homogeneous and predictable, and typically include data acquisition, particle path reconstruction from patterns of activated detector elements, and production of simulated data used to derive detector efficiencies and investigate systematics.

Individuals or groups of physicists searching for specific statistical patterns characteristic of a physics process of interest usually examine the *reconstructed data* from these production jobs. A typical analysis might include preliminary, exploratory data analysis, followed by a process of tuning the analysis code with simulated data. Once physicists refine the analysis procedure, they must determine that procedure's systematics and efficiency, usually through a combination of

1521-9615/08/\$25.00 © 2008 IEEE
Copublished by the IEEE CS and the AIP

ANDREW DOLGERT, LAWRENCE GIBBONS, CHRISTOPHER D. JONES,
VALENTIN KUZNETSOV, MIREK RIEDEWALD, DANIEL RILEY,
GREGORY J. SHARP, AND PETER WITTICH
Cornell University

simulated and real data (avoiding the signal region of interest in real data to avoid biasing the result). Next, they search for the sought-after signal in the data, using their efficiency estimates to derive the size of the signal or an upper bound. Most of these analysis steps are highly iterative and can require months or even years to complete, while the physicist explores different strategies for enhancing the ratio of the physics process of interest to other processes that might mimic it.

Traditionally, a physicist will develop a new strategy during such exploration, perform the analysis using this strategy, plot the distributions, and then finally record the changes along with a printout of the distributions into a physical logbook. The physicist uses this logbook as a reference while preparing presentations or notes for internal distribution within a collaboration, which often elicits suggestions for additional strategies. Eventually, the analysis converges on a result and is documented in internal notes and an eventual paper for publication, both of which are then subjected to a grueling internal review process to assess the result's quality, reliability, and reproducibility. When questions arise during this review process, the lab notebook remains the ultimate reference for how the physicist performed the analysis.

In the past, the reconstructed data sets were typically small enough that the task of extracting physics from them was manageable for individuals or small teams with modest computing resources. These final steps were unscripted, unpredictable, and highly individual, and, as such, depended critically on the individual researcher and his or her ability to rapidly and reliably modify small steps in complex chains of calculations. In fact, the difference between success and failure could often be traced to the researcher's ability to understand these data sets. Among the difficulties typically encountered were problems with precisely determining how intermediate quantities were calculated or the inability to repeat a process when additional reconstructed data (or data reprocessed with an improved understanding of the detector) became available.

In the LHC era, this landscape has changed. With the advent of copious high-performance computing resources in a global grid environment, each researcher has access to enormous amounts of processing power. The LHC will produce very large data sets; even the data sets that small teams of researchers will analyze will be terabytes in size. The job of managing these large samples and the ability to explore the data's full possibility will quickly overwhelm the traditional tools that sci-

entists use to manage their workflow: notebooks, scripts, and the scientist's own memory.

At the same time, the increasingly common practice of regularly reprocessing all of an experiment's collected data with the latest version of reconstruction software, along with the growing application of statistically unbiased analysis techniques, have increased the demand to regularly replicate an analysis procedure. These requirements also increase the maintenance costs of using private subsets of the data outside the experiment's official production system, a conflict that can result in analyses not using all the available data or even the best version of it.

LHC experiment participants recognize the central role that the systematic collection and use of data provenance must play in establishing the quality of their results, and some have made attempts at applying modern workflow management technologies to the task.¹ However, at least in our experience on the CMS experiment, these efforts tend to concentrate on the relatively well-understood production use cases, not the more challenging analysis tasks.

CLEO-c EventStore

We designed and implemented the EventStore system² for the medium-energy CLEO-c experiment in the course of migrating away from the commercial object database used to store data in the CLEO III experiment. The project's principal goals were to provide a simple method for physicists to reliably and reproducibly access a consistent set of reconstructed data for analysis. Because the CLEO-c experiment reused much of the software and analysis methodologies from CLEO III, we already had well-developed use cases for the data analysis's later steps to guide our design.

EventStore is primarily a metadata management layer that organizes data for analysis into grades that represent phases in the data management life cycle. Examples of data grades include *raw*, *reco*, and *physics*, which represent raw data directly from the detector, reconstructed data not yet validated for physics use, and reconstructed data approved for physics analysis, respectively. The *physics* grade also includes any corrections calculated after the primary reconstruction step, so physics analysis jobs don't access auxiliary data sources—instead, a physicist selects a data set simply by specifying the grade and a date (usually the date on which the analysis began), along with any other selection criteria. The date brings up the internally consistent view of the data that was current on that date, guaranteeing the analysis's reproducibility

and consistency as long as the same date is used throughout the analysis. The result is a simple, easy-to-use system that meets the most common requirements for consistency and reproducibility.

The EventStore implementation associates a *specific version label* with data derived in a production job. This label identifies the process that produced the data, the CLEO-c software suite's release version, and the process configuration. An EventStore grade includes a set of directed acyclic graphs of specific version labels, thereby recording the set of data that comprises the grade as well as the data's lineage. In our initial implementation, we set the specific version label administratively in each production job's configuration file, but we soon realized that we could make the system more reliable and potentially more useful if labels were generated automatically, so we developed a method based on a hash of the process configuration. This specific version hash is written to a record type that our framework normally propagates from input files to output files, so each output file automatically accumulates a full processing lineage. Later still, we modified the hash computation module to also store the full process configuration in the output record, so that a physicist could replicate a job's configuration even if the original configuration wasn't version controlled or was otherwise lost.

We designed the CLEO-c EventStore with three distinct implementations, tailored to the application's scale: personal, group, and collaboration. The *personal* EventStore is intended to manage data for a physics analysis project on a personal system such as a laptop or desktop. The *group* EventStore is designed to manage substantial amounts of data on a pool of disks, typically at a second-tier analysis facility. The *collaboration* EventStore adds more sophisticated data management facilities to the group system, including support for hierarchical storage, data replication and relocation, and a distributed directory service. The collaboration-scale EventStore resides at the experiment's hosting institution and holds the master copies of all CLEO-c data.

Although our design focused on analysis use cases, we unexpectedly found that the personal EventStore was also very useful during organized production processes, so in the current CLEO-c production workflows, all results are stored in a personal EventStore specific to a particular job. The production team commits the workflow's final results into the collaboration EventStore only after the data pass all validation tests. Unfortunately, CLEO-c didn't widely adopt the personal

EventStore for the analysis use cases for which we designed it, perhaps in part because the CLEO experiment's physics analysis workflows were well established by the time we introduced the EventStore system for CLEO-c data.

CMS Dataset Bookkeeping Service

The CMS Dataset Bookkeeping Service (DBS)³ is the primary metadata repository for the CMS experiment. It is conceptually similar to the EventStore's metadata management components but differs in scale and implementation. With thousands of jobs at hundreds of institutions accessing CMS data at any time, scalability and reliability are much more challenging, and the design and implementation involved a much larger group drawn from many institutions within CMS.

The CMS DBS is designed as a hierarchical federation, with a global instance hosted at CERN's experiment site and local instances used for intermediate results of production jobs. As in the EventStore system, production jobs record their final results in a local DBS instance. The production team validates the results and, if validated, publishes them to the global DBS. However, unlike in CLEO-c, all operations in the CMS system are performed on the local DBS instance (including locating the data to be processed), so the CMS workflow management tools must prepare a local DBS by copying metadata about the input data sets into it prior to data processing. This step eliminates any dependence running jobs have on the global DBS, which is a more robust configuration for a globally deployed service.

The DBS records the parentage relationships between a task's input and output files, the configuration of the job that processed the data, and the release version of the CMS software suite used. In the terms of a widely cited taxonomy,⁴ the DBS data provenance is *process oriented*—it collects provenance about the deriving processes with file granularity.

The CMS workflow management tools access both global and local DBS instances via Web services interfaces, while physicists principally browse them in the DBS Data Discovery Web application, written as part of this project.

CMS Data-Processing Software Framework

The provenance collected in the DBS generally isn't sufficient to fully understand how physics analysis results are produced, at least in part due to the CMS data-processing framework's flexibility. The primary CMS data-processing appli-

```

Module: siStripClusters Reco                                # module name and processing step
PSet id:404e226632c0b4f66e9dccb2f7970a46                # configuration ID
products: { SiStripClusterSetsiStripClusters_Reco.      # types produced by this module
}
parameters: {                                             # module configuration
  @modulelabel: string = 'siStripClusters'
  @moduletype: string = 'SiStripClusterizer'
  ChannelThreshold: double = 3
  ClusterThreshold: double = 5
  MaxHolesInCluster: int32 = 0
}

```

Figure 1. A simplified annotation for an object produced by a module that finds clusters of activated channels in a silicon strip detector element.

tion, developed by a large team with members from several institutions within CMS, consists of a single C++ software framework with a suite of plug-in modules.⁵ A framework job's configuration specifies a set of these modules to load, each module's parameters, and a set of processing *paths* specifying the sequence in which the modules are executed. With different configurations and modules, we can use the same framework to collate detector-produced data, generate simulated data, reconstruct particle tracks from patterns of activated elements in the detector, and perform most of the steps in the physics analysis. A single framework job can simultaneously write several different output files with different object contents and filtering criteria; it can also read multiple input files with slightly different processing histories. Understanding in detail how an output file's contents were produced requires finer granularity provenance than the process-oriented DBS provenance.

Moreover, processing steps can also occur outside the official job execution services that record provenance in the DBS, in which case, the process-oriented provenance could be incomplete. It's highly desirable to be able to reliably detect when this has happened. To address such cases, the CMS data-processing software framework implements fine-grained object annotations to record the processing history that produced each object. This framework data provenance is *data oriented*, explicitly recording data products' lineage with the granularity of individual atomic storage objects. To generate the annotations, the framework normalizes the process configuration into a set of declarative statements for each module, which reside in the output file and can be queried from within the framework. The sim-

plified annotation shown in Figure 1 would be attached to objects produced by a module that finds clusters of activated channels in a silicon strip detector element.

The framework mediates all access to the data, which lets it associate all module-produced data with the module configuration and the types of the data the module accessed. These associations among the objects produced, input objects, and the module's configuration are stored in the output file as fine-grained object annotations. Continuing our example, the framework would annotate a set of clusters produced by the `siStripClusters` module with the module configuration displayed in Figure 1, along with the types of objects the module accessed.

Annotations in the input files pass through to the output file, so we can determine an object's complete processing history by recursively following each input object's configuration annotations. This history produces a directed acyclic graph of input data types connected by module configurations, and because the framework doesn't allow a module's configuration to change during the course of a single job, the representation of these annotations can be quite compact.

For organized production jobs, the framework is normally configured to enforce the processing history's homogeneity for all the data in a set by examining the object types stored as well as the object annotations. When it finds homogeneity exceptions, examining the object annotations identifies the difference in processing history and, in our experience, quickly leads to the identification of the programming or configuration error that caused the problem. Once the production team identifies the cause of a problem, they can search the job configuration and software release provenance in the

DBS to identify additional data likely to be affected. Exact homogeneity is also the normal case for the later steps in data analysis, but this constraint is sometimes undesirable. The data-oriented framework provenance makes it possible to detect when strict homogeneity has been violated and still determine exactly how each object was produced.

A subtlety of the software framework's data provenance is that it records the data inputs and calculations that produced an object, but it doesn't record any filtering steps that discard data objects. As a result, the framework provenance can help users understand an individual object or record's processing history, but it might not be sufficient to replicate the entire data set's production process because it doesn't record all the filtering applied. One notable exception is the classification criteria applied by the experiment's *trigger*, which forms the primary basis

The lack of selection criteria is an obstacle to fully understanding an analysis strictly from the framework's provenance.

for dividing the data into data sets. However, the mechanism used to record the trigger classification doesn't fully record logical combinations of trigger classifications and hasn't been generalized beyond recording the trigger decision.

The lack of selection criteria is an obstacle to fully understanding an analysis strictly from the framework's provenance. Another obstacle is that a physics analysis often involves modules private to a group or individual. The CMS collaboration is attempting to standardize a set of higher-level objects that are produced by the organized production process, but any specific physics analysis will have additional steps afterward. One approach to this problem is to develop a set of small, generic analysis components that allow a physicist to construct an analysis procedure entirely via the framework configuration. As a simple example, we could select collision events that contain a track with transverse momentum greater than 15 GeV/c with the following module configuration:

```
module highPtTracks = TrackSelector {
  InputTag src = ctAnalyticalTracks
  string cut = "pt > 15.0"
}
```

where the "src" `InputTag` specifies the object col-

lection to which to apply the selection, and the "cut" string specifies the selection criteria. The module parses the "cut" string by using the Reflex C++ introspection system to locate the object method to which to apply the selection criteria and to construct an efficient implementation of the selection. A physicist typically uses one of these modules to produce a new collection of objects with the selection criteria applied, so that the selection criteria are recorded in the annotation provenance when the framework registers the production of the new collection.

With the generic component system, we can directly read the selection criteria and algorithms applied from the framework annotations. We can also discover how two analyses differ, or check for errors (such as applying a narrow selection criteria in one step followed by a wider selection on the same parameter in a later step) by comparing the provenance of each. How generally useful this system will be for physics analysis is still an open question, particularly when physicists use multivariate classification techniques that can't be configured with just a few parameters.

Replacing the Notebook

Not all physicists use software versioning systems and sound software release practices as consistently as they should for their own work, so modifying past workflows (such as by replicating an old workflow with recently acquired data) can be error-prone because the only records might exist in a physical notebook. We're working on two complementary projects to investigate how existing scientific workflow management tools could replace the physical notebook for HEP data analysis. A secondary objective of this work is to evaluate the feasibility of combining domain-specific CMS provenance with existing scientific workflow tools, which we believe have been underutilized in particle physics.

In 2003, the LHC Grid Computing Project High-Energy Physics Common Application Layer Requirements Technical Assessment Group examined common use cases for data analysis. They identified the desirability of an electronic bookkeeping tool or *electronic logbook* with the following features:

- Every task whose output is retained should be recorded with the output's lineage and sufficient information to replicate the task.
- The logbook should be capable of submitting tasks to the experiment's workflow management system, query the execution status, and automatically record the results of task execution.

- The logbook should make it easy to repeat a task with configuration variations.
- The logbook should be able to report a task's resource consumption.
- The logbook should be usable by individuals as well as groups of physicists. It should be possible to “cut and paste” parts of one logbook into another.
- A group's logbook should be usable concurrently by several group members.

We propose using a local DBS instance as an electronic logbook's primary memory, tailored to track the lineage of all data in a physicist's analysis via much the same mechanisms used for organized production tasks. Using the DBS as the provenance-tracking mechanism for an analysis would also let us reuse the existing CMS workflow management and job execution systems, which the logbook could access via a Web service. Because the CMS workflow management tools normally incorporate the metadata for all input files into the local DBS instance used for a process, it naturally supports such operations as incrementally adding data to an analysis. In addition, the logbook could automatically enforce proper version control of all source code—in fact, recent distributed version control systems would allow synchronization of the source code with a master repository while also supporting full version control even when the master repository is unavailable.

We're currently evaluating the suitability of existing scientific workflow management tools, such as Kepler⁶ and VisTrails,⁷ as an electronic logbook component for managing tasks such as process configuration, replicating a completed task on a new data set, or repeating a task with specific variations in the configuration. In parallel with the electronic logbook project, our HEPTrails project is investigating the strategies used during a physics analysis's exploration phase, independent of the details about a particular experiment's software or computing systems. The implementation of HEPTrails is based on the VisTrails package, which is designed to track the provenance of workflows developed for interactive visualization of scientific or medical data, letting users edit their workflows and then run them.⁸ The system displays the results in a separate window via a spreadsheet metaphor, and each workflow modification is recorded so that users can graphically see the change history along with modifications leading to related workflows. Users can even choose a workflow from a provenance graph and either rerun or modify it to create a new workflow that

will also be recorded in the provenance graph. The provenance capture, along with the ability to use a graph to choose older workflows for modification, are ideal for the HEPTrails project, but the way workflows are executed in VisTrails isn't a good match for HEP data processing.

Specifically, VisTrails decomposes a workflow into modules, with one module's output connected to the input of one or more subsequent modules. When the workflow executes, each module executes once in the order the workflow specifies. In HEP data, each event record is independent of all other event records, so an HEP analysis applies the same workflow to each event, accumulating a summary of all the processed events' properties. However, event records also hold additional data collections (such as lists of particle trajectories or clusters of energy deposited in the surrounding calorimeter), and physicists normally apply the same workflow for all items in a collection. The standard VisTrails workflow execution engine doesn't support this type of repetitive application of a workflow, but its design does support replacing the standard execution engine with one that—along with a set of custom workflow modules—generates a program to implement the desired workflow.

A distribution module can take a collection as input and pass individual elements of that collection to child modules, which can further transform or filter elements. Another form of module monitors the individual items and, when the collection is complete, performs an operation and passes the result to its child modules, allowing implicit iterations to be performed in the workflow. Figure 1 shows an example HEPTrails workflow with implicit looping, filtering, and transformation. These modules support streaming workflows that process data as it becomes available and, in turn, make the data available for further processing as soon as an object or collection is complete.

Because each module's individual operations tend to be very fine-grained, the module's communication overhead can be greater than the operations themselves. To make the workflow execution sufficiently efficient, the modules build an expression tree instead of directly performing an operation. Execution of the workflow transforms this tree into code in the Python language, which is then run on a separate thread. Running the generated code on a separate thread lets a physicist see partial results from long-running workflows and create a new workflow inspired by them while the original workflow runs.

Transforming a workflow to create an intermediate representation for subsequent execution is a

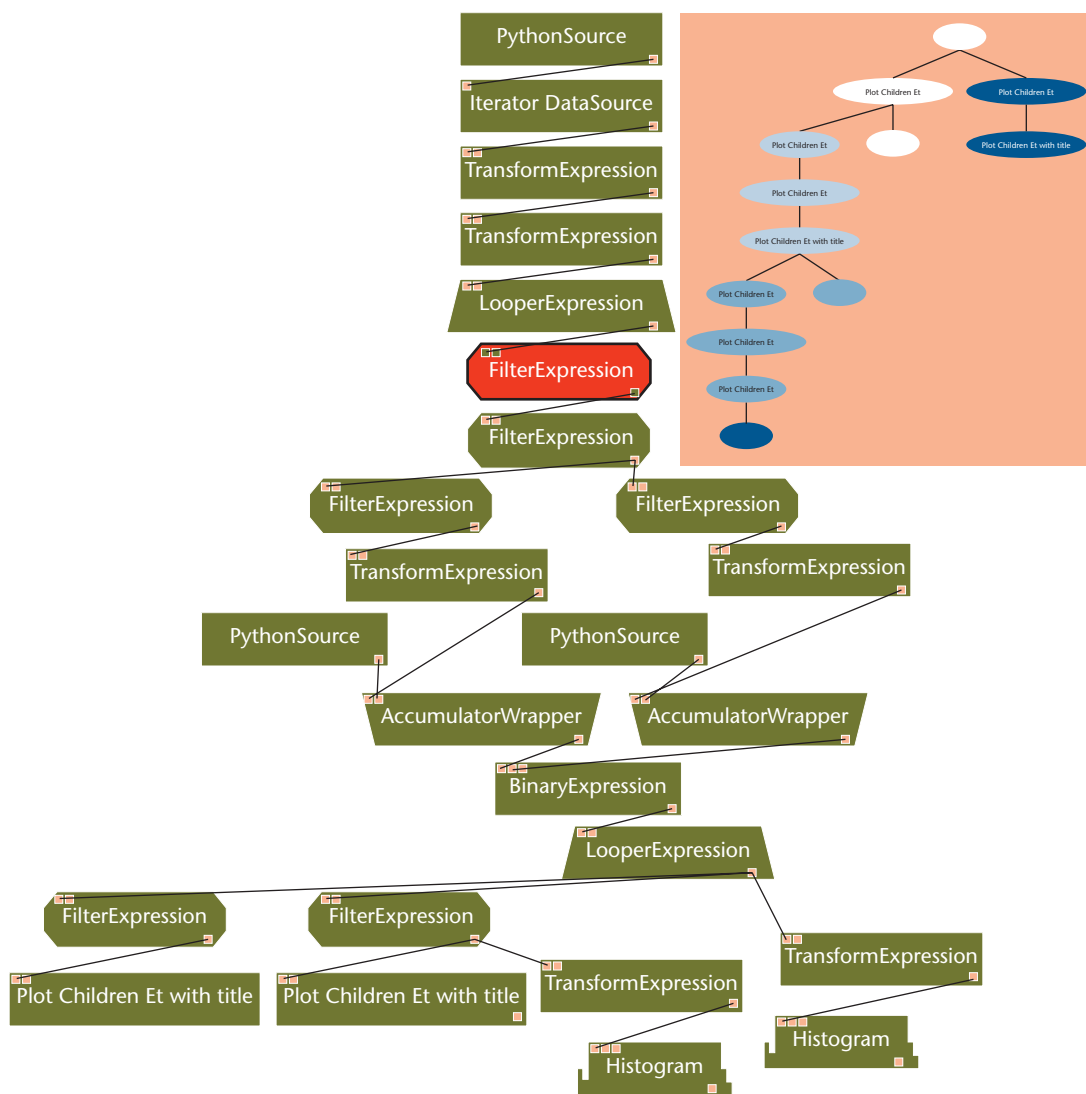



Figure 2. Illustration of a HEPTrails workflow that has implicit looping via `LooperExpression` modules as well as filtering and transformation operations. The inset window shows the workflow’s version history.

general technique with many uses. For example, we plan to use VisTrails to construct configuration files for the CMS software framework and send them to the CMS grid workflow system for execution. In this scheme, the workflow in VisTrails represents the module configurations and execution paths, which—upon execution of the workflow—translates into a framework configuration.

If a physicist modifies a workflow, only the portions of the workflow that need to be re-executed are the parts directly affected by the change or descendants of a changed part. We plan to implement caching of the workflow’s intermediate results and, instead of executing the direct ancestors of a changed part, HEPTrails will replay the ancestry back to only the closest ancestor that has cached the

appropriate intermediate results. HEPTrails will also check that a filter applied late in a workflow isn’t a less restrictive selection criterion than a related filter earlier in the workflow or in a previous one.

The LHC experiments’ scale, collaboration sizes, and sizes of the physics topic subgroups within collaborations are placing new demands for provenance tracking on HEP software design. The CMS collaboration is exploring several approaches to provenance, two of which we’ve discussed here. Although these efforts are largely ad hoc, they do explore interesting design points, particularly with respect to the kinds of prov-

enance collected and the granularity of the data objects to which the provenance refers. The utility of provenance data increases with its availability, but there is a trade-off between provenance data's completeness and the overhead of collecting, storing, accessing, and managing it. We believe the adoption of ubiquitous, decentralized provenance creates new opportunities to better manage the unpredictable, unscripted steps of physics analysis, especially on the vast quantities of data generated from the LHC experiments. 

Acknowledgments

The CMS DBS and CMS data-processing framework sections of this article summarize the work of many contributors within the CMS collaboration, too numerous to recognize individually here, but we particularly acknowledge the contributions of Lee Lueking and Elizabeth Sexton-Kennedy. This work was partially supported by the US National Science Foundation award PHY-0654198.

References

1. A. Arbree et al., "Virtual Data in CMS Production," *Proc. Int'l Conf. Computing in High-Energy Physics and Nuclear Physics (CHEP 03)*, CERN, 2003; <http://arxiv.org/abs/cs.DC/0306009>.
2. C.D. Jones et al., "EventStore: Managing Event Versioning and Data Partitioning Using Legacy Data Formats," *Proc. Int'l Conf. Computing in High-Energy Physics and Nuclear Physics (CHEP 04)*, CERN, 2004; <http://indico.cern.ch/material/Display.py?contribId=199&sessionId=6&materialId=paper&confId=0>.
3. A. Afaq et al., "The CMS Dataset Bookkeeping Service," *Proc. Int'l Conf. Computing in High-Energy Physics and Nuclear Physics (CHEP 07)*, CERN, 2007; <http://indico.cern.ch/material/Display.py?contribId=325&sessionId=28&materialId=paper&confId=3580>.
4. Y. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in e-Science," *SIGMOD Record*, vol. 34, no. 3, 2005, pp. 31–36.
5. C.D. Jones et al., "The New CMS Event Data Model and Framework," *Proc. Int'l Conf. Computing in High-Energy Physics (CHEP 06)*, CERN, 2006; <http://indico.cern.ch/getFile.py?access?contribId=242&sessionId=3&resId=0&materialId=paper&confId=048>.
6. I. Altintas et al., "Kepler: An Extensible System for Design and Execution of Scientific Workflows," *Proc. 16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM 04)*, IEEE CS Press, 2004, pp. 423–424.
7. S.P. Callahan et al., "VisTrails: Visualization Meets Data Management," *Proc. ACM SIGMOD*, ACM Press, 2006, pp. 745–747.
8. J. Freire et al., "Managing Rapidly-Evolving Scientific Workflows," *Proc. Int'l Provenance and Annotation Workshop, LNCS 4145*, Springer, 2006, pp. 10–18.

Andrew Dolgert is a consultant at the Cornell University Center for Advance Computation. His research interests include data visualization and software usability. Dolgert has a PhD in physics from the University of Virginia. Contact him at ajd27@cornell.edu.

Lawrence Gibbons is a professor in the Department of Physics at Cornell University. His research interests include precision measurement of CKM matrix elements, origin of electroweak symmetry breaking, and analysis tools for large-scale data sets. Gibbons has a PhD in high-energy physics from the University of Chicago. Contact him at lkg@mail.lepp.cornell.edu.

Christopher D. Jones is a senior research associate in the Laboratory for Elementary-Particle Physics at Cornell University. His research interests include software usability, analysis tools for HEP, and data visualization. Jones has a PhD in high-energy physics from Cornell. Contact him at cdj@mail.lepp.cornell.edu.

Valentin Kuznetsov is a research associate in the Laboratory for Elementary-Particle Physics at Cornell University. His research interests include data management, data mining, and Web technologies. Kuznetsov has a PhD in physical and mathematical science from the Joint Institute for Nuclear Research, Dubna, Russia. Contact him at vk@mail.lepp.cornell.edu.

Mirek Riedewald is a research associate in the Department of Computer Science at Cornell University. His research interests include data management and analysis services for the sciences, data stream processing, and data mining. Riedewald has a PhD in computer science from the University of California, Santa Barbara. Contact him at mirek@cs.cornell.edu.

Daniel Riley is a research associate in the Laboratory for Elementary-Particle Physics at Cornell University. His research interests include reliable distributed systems, network security protocols, and analysis tools for large-scale data sets. Riley has a PhD in high-energy physics from Cornell. Contact him at dsr@mail.lepp.cornell.edu.

Gregory J. Sharp is a senior programmer/analyst in the Laboratory for Elementary-Particle Physics at Cornell University. His research interests include operating systems, storage, and theology. Sharp has an ME in computer science from Cornell. Contact him at gregor@mail.lepp.cornell.edu.

Peter Wittich is a professor in the Department of Physics at Cornell University. His research interests include elementary particle physics, hadron collider physics, and neutrino physics. Wittich has a PhD in high-energy physics from the University of Pennsylvania. Contact him at pw94@cornell.edu.