POSTER
Automatic Provenance Collection and Publishing in a Science Data Production Environment—Early Results[⋆]

James Frew[1], Greg Janée[2], and Peter Slaughter[2]

[1] Bren School of Environmental Science and Management,
University of California, Santa Barbara
`frew@bren.ucsb.edu`

[2] Institute for Computational Earth System Science,
University of California, Santa Barbara
`{gjanee,peter}@icess.ucsb.edu`

Abstract. The Earth Systen Science Server (ES3) system transparently collects provenance information from executing code. Provenance information (ancestors or descendants) for any process or data granule may then be retrieved from a web service, in both textual and graphical formats. We have installed ES3 in a quasi-production environment, wherein multiple Earth satellite data streams are synthesized into daily grids of global ocean color parameters, and the resulting data granules published online. ES3's non-intrusive nature makes its insertion into such an environment fairly straightforward, but considerations such as collating distributed provenance (from processes spread across computing clusters) and sharing unique identifiers (to link programs and data granules with their separately-maintained provenance) must still be addressed. We present for discussion our preliminary results from assembling such an environment.

1 Introduction

UCSB's Ocean Color Research Group (OCRG)[3] creates and distributes a variety of ocean color data products as candidate NASA Earth science data records (ESDRs). These products range from ocean optical properties and phytoplankton functional groups to phytoplankton growth rates and carbon-based productivity. Many of the products are derived by merging data from multiple satellite sensor systems [7].

All OCRG products have associated quality indices, which comprise both the statistical confidence of individual values, and how the product as a whole relates to previous versions, similar products, and *in situ* validation data. Additionally,

[⋆] supported by NASA cooperative agreements NNG04GC52A and NNX08AP36A

[3] `http://wiki.icess.ucsb.edu/measures`

OCRG is charged by its cooperative agreement with NASA to track and manage algorithm and data lineage throughout the product generation process, and to implement methods for automatically informing users of updated products or analyses.

ESDRs represent a blurring of the historic distinction between data creators and data providers [2]. Research groups like OCRG are now charged with distributing their data products to the broader research community, as well as developing the science-based algorithms that drive the products' creation.

2 A Non-Intrusive Science Data System

OCRG is part of a larger ocean color science community that has its own computational idioms, artifacts, and traditions. While the underlying phenomena are continuous in space and time, the data products are typically generated and exchanged in fixed granularities, based on both natural units (e.g., daily/weekly/monthly aggregations) and historic limitations (e.g., file sizes small enough to transfer expeditiously over slow connections.) Data granules are usually pre-calculated and managed as files or database BLOBs. Granules typically have standard naming conventions that encode significant product semantics (e.g., dates, regions, version numbers) and are distributed in complex formats (e.g., HDF, netCDF) that commingle metadata and multidimensional data.

The ocean color science community has additionally standardized on a specific computational environment (SeaDAS[4]) based on an interpreted array manipulation language (IDL[5]). Data product generation occurs either manually (by invoking specific commands) or quasi-automatically (by batch jobs.) Data publication involves simply moving the product files onto a web server. High-level metadata (e.g., publications describing the algorithms) is available, but supplied separately.

Introducing product and provenance management into such an environment is challenging. We have simplified this problem by adopting a *microservices* approach. Microservices [1, 8] are an architectural pattern, originally developed in the digital curation community, in which a system's functionality is devolved into small, self-contained, interoperable services. While we support a comprehensive array of microservices for data product search, access, and metadata management, we focus here on those associated with identity and provenance management.

3 Identity Management

Persistent identification of datasets, data granules, external metadata, and ancillary resources is critical to maintaining the documentation and reproducibility that are the hallmarks of the scientific method. Strictly speaking, persistence of

[4] `http://oceancolor.gsfc.nasa.gov/seadas`
[5] `http://www.ittvis.com/idl`

identifiers over time is an outcome–a result of commitment–but the technical forms of identification can make persistence more or less difficult to achieve.

For data products that are released in discrete, named versions, such as OCRG products, incorporating versions in identifiers is an additional challenge. Specifically, version-aware identifiers must allow users of a granule to:

- Cite a specific version of a granule;
- Cite the most recent version of a granule, and/or a granule *sans* version; and
- Detect that a version of a granule no longer exists, and be directed to appropriate metadata and from there to a newer version.

We are developing an approach to version-aware identity management that satisfies these requirements and that is built on two technologies: well-known persistent identifier schemes such as Archival Resource Keys (ARKs) [6], and HTTP redirection. A granule identifier in this approach consists of a dataset identifier drawn from a persistent identifier scheme, to handle dataset-level relocation over time, prefixed to a filename that includes a version indicator and other semantics. HTTP redirection rules, in the form of regular expressions, express version defaulting and granule deletion. The net result is an identity management system that requires almost no cooperation from the data providers or data production system.

Our preliminary work on integrating version-related redirections with persistent identifiers has led us to select ARKs as the most appropriate persistent identifier scheme. Unlike Digital Object Identifiers (DOIs)[6], ARKs allow qualifying information (e.g., granule identifiers) to be appended to the dataset identifier and passed along by the identifier resolution system. Unlike Persistent URLs (PURLs)[7], ARKs are self-identifying, and don't require the reservation of a portion of the HTTP URL namespace for their implementation.

4 Provenance Management

The ES3 system [3] collects provenance information from executing code, using a combination of system call tracing, transparent wrapping, and application environment instrumentation. For clarity, we will limit the discussion hereafter to ES3's system call tracing ("shell plugin") mode. In this mode, provenance collection entails:

1. The ES3 **collector** process is started on the processing host system. The collector waits for provenance event messages from an instrumented process.
2. A science process is invoked on the processing host system, from an ES3-instrumented command interpreter (usually `bash`.) The science process may be arbitrarily complex, and is usually a script that invokes several other processes. The science process itself is not modified in any way, and the ES3 command interpreter behaves identically to a standard one; this is why we call ES3 "non-intrusive."

[6] `http://doi.org`
[7] `http://purl.org`

3. The ES3 command interpreter sends traces of all the science process's system interactions to the ES3 collector, which formats them, discards unwanted detail, and saves this "raw provenance" to a log file. Raw provenance is simply a set of tuples of the form
   (*process ID*, *timestamp*, *system call*, *arguments* [, ...])
   where the system calls are limited to "provenance events;" i.e., file access, process creation, or program execution. The collector does some simple editing of the raw provenance (e.g., converts file descriptors to corresponding file names) consistent with near-real-time processing.
4. When the science process exits, the ES3 **transmitter** process is invoked and scans the log file. The transmitter assembles the raw provenance into a provenance graph and submits the graph components to the ES3 database, as follows:
   (a) Provenance graph nodes are created by assigning an automatically generated unique identifier (UUID) to each provenance event.
   (b) Any filesystem object referenced by a provenance graph node and readable by the transmitter is checksummed with a secure hash algorithm (e.g., SHA-1[8]). This checksum is included in the provenance metadata.
   (c) Connected events (e.g., processes writing to and reading from the same file or pipe) are indicated by creating a provenance graph edge between the appropriate UUIDs.
   (d) The provenance graph nodes and edges are formatted as XML messages and sent to the ES3 database.

The ES3 database allows event identifiers to be queried for their associated metadata (date, time, host system parameters, etc.), and of course for their parent and/or child events. The parent/child queries may be recursive, generating forward and/or reverse provenance to any specified depth. Provenance metadata is delivered as serialized graph in XML.

ES3 currently provides post-processors that convert the ES3 native provenance graph format to GraphML[9] or DOT[10], for visualizing in tools such as yEd[11] or Graphviz[12]. However, we realize that different kinds of queries or user communities may require alternative provenance renderings [5]. We are therefore exploring connecting the ES3 database to a generic web-based graph browsing system [4].

5 Issues Raised

A meta-issue we address is the need to manage multiple kinds of identifiers–datasets, granules, provenance events–and to make the mappings between them as transparent as possible.

[8] `http://www.ietf.org/rfc/rfc3174.txt`
[9] `http://graphml.graphdrawing.org`
[10] `http://www.graphviz.org/doc/info/lang.html`
[11] `http://www.yworks.com/en/products_yed_about.html`
[12] `http://www.graphviz.org`

ES3 has been developed in a cluster computing environment, which is a primary reason that it uses a decentralized event identifier scheme (UUIDs.) ES3 transmitters running on multiple hosts can thus submit provenance information to a more centralized (e.g., per-cluster) database without danger of identifier collisions.

Note, however, that the mapping between the provenance *events* recorded by ES3 and the *objects* managed by the rest of the data system is many-to-one: any object may participate in many provenance events. Likewise, the mapping between object identifiers and checksums can be many-to-one, since objects may be updated. The ES3 database therefore supports queries against these mappings (e.g., return an objects's provenance events, given its checksum.)

The identity management service, on the other hand, is concerned with mapping *published* names (persistent identifiers) to the appropriate internal objects. Provenance management enables the partial or complete automation of this mapping by allowing concepts like "product" and "version" to be functionally defined—for example, a particular version of a dataset might defined as all data objects generated by a specific instance (as defined by a checksum) of a particular algorithm (as defined by a filename.)

We are examining how to best incorporate and advertise published identifiers in metadata, and in the granules themselves.

References

[1] S. Abrams, J. Kunze, and D. Loy. An emergent micro-services approach to digital curation infrastructure. In *Proceedings of the Sixth International Conference on Preservation of Digital Objects*, San Francisco, CA, October 2009.

[2] J. Frew and R. Bose. Lineage retrieval for scientific data processing: A survey. *ACM Computing Surveys*, 37(1):1–28, March 2005.

[3] J. Frew, D. Metzger, and P. Slaughter. Automatic capture and reconstruction of computational provenance. *Concurr. Comput. : Pract. Exper.*, 20(5):485–496, 2008.

[4] B. Gretarsson, S. Bostandjiev, J. O'Donovan, and T. Hollerer. WiGis: a scalable framework for web-based interactive graph visualizations. In D. Eppstein and E. R. Gansner, editors, *Graph Drawing: 17th International Symposium, (GD 2009)*, volume 5849 of *Lecture Notes in Computer Science*, pages 119–134, Chicago, IL, September 2009. Springer Berlin / Heidelberg.

[5] M. Kunde, H. Bergmeyer, and A. Schreiber. Requirements for a provenance visualization component. In J. Freire, D. Koop, and L. Moreau, editors, *Provenance and Annotation of Data and Processes*, volume 5272 of *Lecture Notes in Computer Science*, pages 241–252, Salt Lake City, UT, June 2008. Springer Berlin / Heidelberg.

[6] J. A. Kunze. Towards electronic persistence using ARK identifiers. In J. Masanès, A. Rauber, and G. Cobena, editors, *3rd Workshop on Web Archives*, pages 4–12, Trondheim, Norway, August 2003.

[7] S. Maritorena and D. A. Siegel. Consistent merging of satellite ocean color data sets using a bio-optical model. *Remote Sensing of Environment*, 94(4):429–440, 2005.

[8] R. Moore. Towards a theory of digital preservation. *International Journal of Digital Curation*, 3(1):63–75, 2008.