Asynchronous Data Provenance for Research Data in a Distributed System

Benedikt Heinrichs[a] and Marius Politze[b]
IT Center, RWTH Aachen University, Seffenter Weg 23, Aachen, Germany

Keywords: Research Data Management, Data Provenance, Distributed Systems.

Abstract: Many provenance systems assume that the data flow is being directly orchestrated by them or logs are present which describe it. This works well until these assumptions do not hold anymore. The Coscine platform is a way for researchers to connect to different storage providers and annotate their stored data with discipline-specific metadata. These storage providers, however, do not inform the platform of externally induced changes for example by the user. Therefore, this paper focuses on the need of data provenance that is not directly produced and has to be deduced after the fact. An approach is proposed for dealing with and creating such asynchronous data provenance which makes use of change indicators that deduce if a data entity has been modified. A representation on how to describe such an asynchronous data provenance in the Resource Description Framework (RDF) is discussed. Finally, a prototypical implementation of the approach in the Coscine use-case is described and the future steps for the approach and prototype are detailed.

1 INTRODUCTION

The FAIR Guiding Principles have evolved to form a guideline for research data management (RDM). Accordingly, data should be findable, accessible, interoperable and re-usable as described by (Wilkinson et al., 2016). In turn RDM plays an increasingly important role in today's university landscape. The idea is to incorporate those principles in every aspect of research where data is being produced and incorporate them into the scientific workflow. While a lot of work has been done to improve the scientific workflow, the different research domains diverge a lot in their approaches. This diversion starts with the choice of a storage provider which with the emergence of cloud storage and technologies like object storage ranges over far more options than just classic solutions like normal file systems. Platforms like Coscine, described by (Politze et al., 2020) and formerly called "CoScInE", or Open Science Framework, described by (Foster and Deardorff, 2017), try to deal with these issues and integrate solutions like WaterButler[1] to make research data accessible across domains and storage providers in one single place. The further use and capability of creating metadata makes the research data more in line with the FAIR Guiding Principles. The main issue however with such an approach is that due to the distributed nature of the system, it is difficult to represent the path the research data took and with that, the data provenance. Classic more centralized workflow provenance systems like Taverna, Pegasus, Triana, Askalon, Kepler, GWES, and Karajan described by (Talia et al., 2013) usually either track the movement of data since this movement is directly orchestrated by them (eager) or they can rely on the provenance already being logged (lazy) as described by (Cruz et al., 2009). This work presents a different approach to encounter the fact that researchers can change research data on a storage provider without the integration platform receiving any notice, breaking the whole provenance model. For this reason, this paper will focus on the generation of data provenance for research data that has been changed by an external impact with no information being recorded or event being sent, here called asynchronous data provenance. In this context the classic approaches are in contrast described as synchronous data provenance methods since there is a clear way to track the movement of data and directly describe its provenance.

[a] https://orcid.org/0000-0003-3309-5985
[b] https://orcid.org/0000-0003-3175-0659
[1] Codebase by Center for Open Science: https://github.com/CenterForOpenScience/waterbutler/

Heinrichs, B. and Politze, M.
Asynchronous Data Provenance for Research Data in a Distributed System.
DOI: 10.5220/0010478003610367
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 2*, pages 361-367
ISBN: 978-989-758-509-8

2 CURRENT STATE AND RESEARCH GOAL

Since data provenance is necessary for reproducing the path data traversed, it is a quite popular field and a lot of research has done in the area. This section will therefore focus on the most important aspects of data provenance in regard to the topic of the paper. An overview on provenance in general will be given and furthermore it will be looked at how different provenance systems are implemented, how provenance currently works in distributed systems and how provenance is represented. After this current state, the discovered challenges for this paper are formulated.

2.1 Provenance

Provenance is defined by the researchers in (Herschel et al., 2017) as "any information describing the production process of an end product". In their survey they define different levels of provenance in a hierarchy. At the top of the hierarchy, data provenance is placed as it looks into the processing of individual data items at the highest resolution. The researchers in (Davidson et al., 2007) and (Davidson and Freire, 2008) furthermore define that provenance can be prospective, e.g. capturing the specification of a workflow, or retrospective, e.g. capturing the executed steps of a workflow. The problem presented in this paper however does not really fit into either of the areas, since here the need for creating provenance after the fact is expressed. Therefore, previous solutions for recording provenance are here defined as synchronous provenance while the one being looked into this paper is defined as asynchronous provenance.

2.2 Current State on Provenance Systems

In the review conducted by (Pérez et al., 2018) relevant provenance systems were evaluated and their common characteristics described. The approaches to tracing data capture are defined as eager and lazy. These existing approaches are however still scoped in a very centralized manner, expecting either a way to directly compute provenance from a workflow or having some kind of logging present to compute the provenance from. Regarding implementing data provenance in systems which do not offer full data provenance yet, examples like the researchers in (Interlandi et al., 2018) including data provenance in Apache Spark show promise. Such solutions are however too specialized and too centralized to use for asynchronous data provenance in a distributed system.

2.3 Current State on Provenance in Distributed Systems

The work described by (Mufti and Elkhodr, 2018) gives an overview on data provenance in the internet of things (IoT), which is distributed innately. It outlines the usefulness and challenges in the area. Furthermore, the researchers in (Hu et al., 2020) look into data provenance in the IoT as well and review existing methods based on general and security requirements. Concrete implementations like the one described in (Smith et al., 2018) and (E. Stephan et al., 2017) show that still some kind of integration in a system is necessary or provenance information needs to be logged to describe the data provenance. Finally, (Ametepe et al., 2018) gives an overview on the current trends, methods and techniques for provenance in a distributed environment.

2.4 Representing Provenance

For representing provenance, some standards exist. In this paper, the focus will be on representing the provenance as linked data in the Resource Description Framework (RDF), described by (Cyganiak et al., 2014). The way for representing provenance in this scope is the W3C standard PROV-O, described by (Belhajjame et al., 2012), which is an ontology that provides classes, properties and restrictions to represent provenance information in RDF. Furthermore, when looking at a data entity a way has to be declared to uniquely reference it. One method to do that is using a persistent identifier (PID) like ePIC, described by (Schwardmann, 2015), since common ways like normal URLs might suffer from problems called "link rot" at some point and identifiers like GUIDs cannot be resolved so easily. PIDs can be efficiently used to connect distributed systems as discussed by (Schmitz and Politze, 2018).

2.5 Challenges

From the current state-of-the-art the following challenges in regard towards dealing with asynchronous data provenance were identified:

- There is a gap for dealing with and producing provenance of data where the change does not produce an event or a log entry
- A way has to be found to describe such asynchronous data provenance items

3 APPROACH

Following the current state-of-the-art and discussed challenges, the proposed approach to deal with them is presented. First the state of the use-case is described. Some requirements collected from there on what amounts to a changed data entity will be discussed further. Derived from that, the structure of the use-case will be adapted to support asynchronous data provenance. Using all these findings, finally the representation proposal of the asynchronous data provenance will be shown.

3.1 Current State of the Use-case

The open-source platform called Coscine[2] was described by (Politze et al., 2020) and (Bensberg, 2020). It supports researchers that want to adhere to the FAIR Guiding Principles by allowing them to access their research data across multiple domains and distributed storage providers. This is further enhanced by giving the researchers the opportunity and guiding them to annotate their research data with so-called metadata which is used as a description of the research data and represented in RDF. Such a solution, however, comes with the issue that research data can be changed externally at any point without the platform being able to take notice on the change and updating the provenance information. Furthermore, coordinated approaches towards versioning of research data or at least their metadata are currently missing and data and metadata movement in between otherwise unconnected storage providers is not tracked. These properties and things to improve upon make Coscine a natural fit for a use-case to implement and apply asynchronous data provenance.
The platform currently stores the metadata for all data entities in an RDF-based knowledge graph uniquely identifying every data entity with a persistent identifier (PID) and overwrites the metadata on every change. The graph name, furthermore, follows the PID syntax, so that the described data entity can be resolved and accessed with a simple URL. To realize this, in the graph name the specific storage provider and the location of the particular data entity is described. Therefore, the structure is defined as: `https://hdl.handle.net/{prefix}/{storage-provider}@path={path}`.

[2]Codebase: https://git.rwth-aachen.de/coscine

3.2 Determining the Occurrence of Change

For asynchronous data provenance, a way has to be established for determining the occurrence of a change to accurately describe it. For this, in the following the existing types of changes and the indicators that can recognize a change will be defined.

3.2.1 Definition of Change Types

Following the discussion, it has to be defined what a change is and what different types of changes exist. The current state-of-the-art focuses a lot on changes which are triggered by the addition or update of a data entity or metadata set and this being directly recognized or logged. Such a kind of change is therefore described as synchronous because it is clear where, when and from whom this change is coming from. For changes where the where, when and from whom are not entirely clear, however, a new definition must be defined. These changes are dependent on external factors like the external addition or update of a data entity or metadata set which are not captured by the integrating platform that wants to describe it. To track them, the previous state has to be compared with the current one at some undefined point after the original change has taken place which is why these changes are defined as asynchronous.

3.2.2 Definition of Change Indicators

For determining the asynchronous data provenance, methods have to be defined which can describe the occurrence of a change. The following list details the available methods from the current use-case.

Modified Timestamp. In general storage providers keep track of the timestamp when a data entity has been modified. This however does not mean that the content really has changed, just that an operation on the data entity has been performed.

Hash. A storage provider can provide a hash of the data entity with an algorithm like SHA-1 which gives a good indicator and identifier of the current version of the data entity and if it has changed. Since a hash is easy to compute and store, this makes this method a fast indicator of change.

Version. Some storage providers can be trusted to give accurate information about the version a data entity holds and this is therefore a clear indicator to represent that a data entity has changed.

Descriptive Metadata. Using the works of (Heinrichs and Politze, 2020) descriptive metadata that describes the content of a data entity can be used to make sure if a data entity has really changed and more importantly show what has changed. This has some implications on the definition of change since e.g. adding a space to a text file would not be recognized as a change because the interpreted content stays the same. Such a method would, therefore, be more focused on content-based (e.g. a changed fact) or structure-based (e.g. different line-count) changes and not the raw changes that previous indicators are built on.

Byte-by-Byte Comparison. For completion, a byte-by-byte comparison could definitely discover if a change has happened. However, storing and comparing many versions of data entities is a very storage and time intensive task and therefore not very favored, especially since previous indicators are much less complex and achieve similar results.

Content-based Domain-dependent Comparison. If a method exists which can produce content-based information from a data entity, this information can be used to figure out differences between two data entities. This however has some limitations because it is very domain-dependent and there is no generalized way to represent that information. The descriptive metadata indicator furthermore is a more defined variation of this and looks therefore superior.

3.3 Creating a Structure which Supports Asynchronous Data Provenance

Following on the definition of the use-case and the definition of change and indicators, a new structure is proposed that supports asynchronous data provenance. First, the versions which have to be kept track of for the data entities and metadata will be defined. Utilizing this information, the new structure is then presented. Furthermore, the integration in the life-cycle of data and their metadata is described.

3.3.1 Definition of Versions

Since data entities and metadata can and should be versioned, a definition on what kind of separate versions in this context exist has to be created. In the Coscine use-case, one thing which would be necessary for a complete implementation of asynchronous data provenance is to store every kind of version in a so-called metadata store to ensure a correct linking between them. The following lists the version types and describes the requirements to ensure this correct linking.

Metadata Version. When a metadata set is created or updated, it has to receive a version. It is furthermore important to link the current version of the metadata set to the previous one.

Data Entity Version. When a change has been detected on a data entity, the data entity should receive a new version which is either generated or received from the storage provider. This version change should trigger an update of the metadata set and the new version of the data entity is stored alongside it.

3.3.2 New Graph Structure

Following the previous definitions, the new graph structure is now presented which should incorporate asynchronous data provenance. The previous structure was adapted in this new concept and improved upon for an easier adoption. The main structure revolves around an overarching graph which contains the provenance information, links to every versioned graph, contains the versions of the data entity and contains optionally additional metadata for the change indicators which have been used. Each versioned graph contains then the metadata for a data entity. The different graphs are a necessity since otherwise the difference between the versions of metadata sets cannot be determined. The distinction is made by an additional parameter in the graph name which defines the metadata version. Such an approach differentiates itself from current version implementations, e.g. done in Zenodo and discussed by (European Organization For Nuclear Research and OpenAIRE, 2013), to keep compatibility with the current graph name and shift the resolving responsibility more to the platform receiving these attributes. Furthermore, for the descriptive metadata a concept is presented which shows how to represent this kind of metadata in the new graph structure. The distinction here is an additional parameter which specifies that the current graph contains extracted descriptive metadata. The graph names for the new concept are shown in the following.

- Overarching Graph:
  `https://hdl.handle.net/{prefix}/{storage-provider}@path={path}`
- Versioned Graph:
  `https://hdl.handle.net/{prefix}/{storage-provider}@path={path}&version={version}`
- Descriptive Metadata Graph:
  `https://hdl.handle.net/{prefix}/{storage-provider}@path={path}&version={version}&extracted`

3.3.3 Integration in the Data Life-cycle

With the new graph structure, it has to be shown that it fits into the life-cycle of data. The steps being looked into are the creation, update and deletion of data and its metadata. Their inclusion and usage is detailed in the following.

Creation. The creation of a data entity enforces the manual or automatic creation of a metadata set. Furthermore, the creation of a metadata set should point to a created data entity. In both cases, the versioned graph for the metadata set is set to "1". The parent graph receives an entry for the new versioned graph and the version for the data entity which is either "1" if not provided or the by the storage provider supplied version.

Update. When a data entity or a metadata set has been updated and a change has been detected, a new versioned graph gets created. This graph increases the last metadata version by "1" and stores the current metadata with it. The parent graph receives an entry for the new versioned graph, links the new one to the old one, stores the version of the data entity and stores the information about the used change indicator.

Deletion. A deletion of a data entity is seen as a special type of update since the metadata set is never deleted, it is only invalidated. The metadata set itself can only be emptied according to the requirements on it, however it can never be deleted. The invalidation is marked in the parent graph.

3.4 Representing Asynchronous Data Provenance

Since the questions where and at which step have been answered, the how question is still open. Therefore, in the following the concrete proposal for representing the asynchronous data provenance will be presented.

As PROV-O is a W3C standard for describing provenance, it was chosen for this approach since the provided terms are detailed enough for this use-case. It is to note that when the term "entity" is being used, in this case a "data entity" is meant and thought of as represented in the previously defined overarching graph and directly linked to the respective versioned graph.

Figure 1 shows the PROV-O representation of an entity being generated at a certain point. This representation is particularly useful in the case of asynchronous data provenance since it can be the case that a new data entity will just be found at an uncertain point with no agent which can be attributed to this data entity. Such a representation therefore can be used in this use-case with a data entity as a PROV-O entity and receiving an asynchronous generation activity with the "xsd:dateTime" data type, representing the time it was discovered.

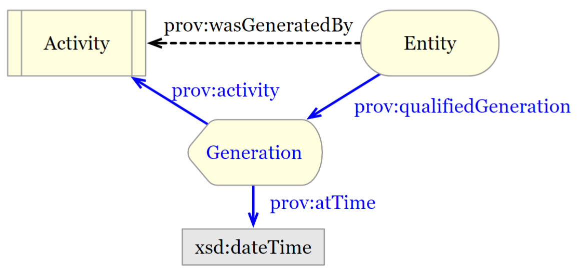

Figure 1: Generating an entity with an activity as defined and visualized by (Belhajjame et al., 2012).

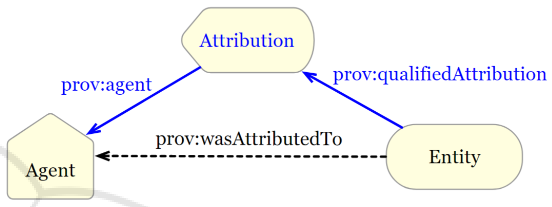

Figure 2: Attributing an entity to an agent as defined and visualized by (Belhajjame et al., 2012).

Following on figure 1, utilizing the concept presented in figure 2 moreover produces an entity that has a generation activity and an agent which is responsible for this entity. This agent can furthermore be responsible for the generation activity if it is linked to the generation activity with the predicate "prov:wasAssociatedWith". This type of attribution to an agent is useful in this context since a data entity can be generated by a platform itself and therefore it is important to distinguish the asynchronous generation activity from the synchronous one. The asynchronous generation activity can however also be associated with an agent since PROV-O provides the capability of describing an "Agent" as a "SoftwareAgent", it just has to be kept in mind that the agent should differentiate from the synchronous data provenance agents so that the difference can still be seen.

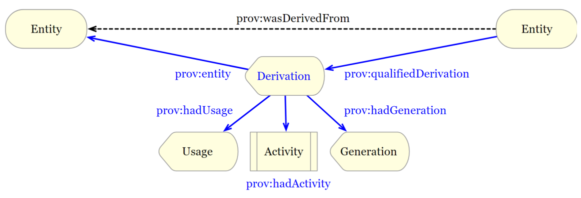

Figure 3: Deriving an entity from another entity as defined and visualized by (Belhajjame et al., 2012).

After an entity can be described, figure 3 focuses on the derivation of an entity from another. This is important in the context of updating a data entity and metadata set for describing the path that entity traversed. It furthermore plays a role in data entities which move across different storage providers. Such an update description is either triggered synchronously or asynchronously by looking at the change indicators and determining that a change has occurred. Both cases result in a new entity description and the linking to the old one shown in figure 3.

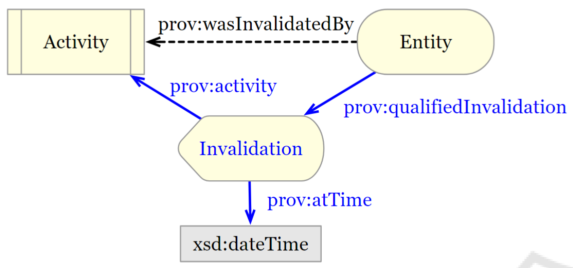

Figure 4: Invalidating an entity with an activity as defined and visualized by (Belhajjame et al., 2012).

Finally, the end of the path a data entity took has to be described. If for some reason, the data entity has been deleted, the representation in 4 can provide a solution for that. The deletion of a data entity therefore does not mean that the provenance metadata and everything related to it will be removed, it just means that the data entity will be marked as invalidated.

4 PRELIMINARY RESULTS

The discussed approach for dealing with asynchronous data provenance has been prototypically implemented for the Coscine use-case. The result is an application that on execution goes over all data entities that are set to be analyzed. When a new data entity has been detected or the defined change indicators (in this case “Hash” and “Descriptive metadata” as defined in 3.2.2) have been triggered, the new entity is represented as discussed in section 3.4. The change indicators are represented by the predicate “foaf:sha1” for the “Hash” change indicator and the entity is marked as “a foaf:Document” as defined by (Brickley and Miller, 2014). Furthermore, following the work from (Heinrichs and Politze, 2020) the descriptive metadata is extracted and stored as discussed in section 3.3.2. The version of the data entity is stored using the predicate of “schema:version” by declaring the entity as “a schema:CreativeWork” as discussed by (Guha et al., 2016). The application is scheduled to run daily, however can be triggered at any point manually, if a change in data entities is expected. For representing the synchronous data provenance, event listeners are used to track the creation, update and deletion of a data entity and a metadata set and the representation is made according to the definitions in section 3.4.

5 CONCLUSION

This paper described the need for a way to capture data provenance in an environment that does not log or produce any data provenance related information since changes are caused by external events like user interactions in a distributed system. The data provenance being established and needed was defined as asynchronous data provenance and a concept was proposed which can represent it. A clear need to describe the type of change for a data entity was established and the indicators for a change were presented. Furthermore, an approach was defined for describing the provenance information in separate graphs and describing the interplay between them. With this, a concept was created to describe the path a data entity has traversed, even if the change is not directly triggered or logged by the integrating platform.

5.1 Identified Research Gaps

Although a concept was presented which can deal with asynchronous data provenance, some questions are still open that need to be answered for completing a full implementation and are presented in the following.

- There are a lot of possible change indicators provided and some of them are used in the prototypical implementation, however a question still remains. Are there any more change indicators out there and which are the best choice for which scenario?
- How can change indicators be evaluated?
- Can machine learning fuel a change indication algorithm and even get a rough estimation on how similar the old and new data entity is?
- How can the current graph structure be generalized to other persistent identifier and prefixes?
- What is a good time frame to try and establish changes on external storage providers?

5.2 Future Work

Following on this work, the prototypical implementation will be enhanced, so that it not only supports the requirements of the Coscine platform, but can be used in general. With work looking into different change indicators and implementing them, it will be seen how well they perform and which produce the best results in performance, accuracy and applicability. Even new developed methods, which could focus on machine learning and fueling a change indicator based on a trained model, could be interesting in this evaluation. Furthermore, the paper is focused on data entities however an interesting area to look into is how to reflect this research on collection of data entities and collection of collections. Since a change indicator is proposed, another area of future interest is to see if the difference between data entities in two collections could tell the similarity between them. Especially by looking into the descriptive metadata as well, the defining topics which are similar could be detected by such a method and most importantly which topics are not the same.

REFERENCES

Ametepe, W., Wang, C., Ocansey, S., Li, X., and Hussain, F. (2018). Data provenance collection and security in a distributed environment: a survey. *International Journal of Computers and Applications*, pages 1–15.

Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. (2012). Prov-o: The prov ontology.

Bensberg, S. (2020). An efficient semantic search engine for research data in an RDF-based knowledge graph. Masterarbeit, RWTH Aachen University, Aachen. Veröffentlicht auf dem Publikationsserver der RWTH Aachen University; Masterarbeit, RWTH Aachen University, 2020.

Brickley, D. and Miller, L. (2014). Foaf vocabulary specification. http://xmlns.com/foaf/spec/.

Cruz, S., Campos, M., and Mattoso, M. (2009). Towards a taxonomy of provenance in scientific workflow management systems. *SERVICES 2009 - 5th 2009 World Congress on Services*.

Cyganiak, R., Lanthaler, M., and Wood, D. (2014). RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

Davidson, S., Cohen-Boulakia, S., Eyal, A., Ludäscher, B., McPhillips, T., Bowers, S., Anand, M., and Freire, J. (2007). Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30:44–50.

Davidson, S. and Freire, J. (2008). Provenance and scientific workflows: Challenges and opportunities. pages 1345–1350.

E. Stephan, B. Raju, T. Elsethagen, L. Pouchard, and C. Gamboa (2017). A scientific data provenance harvester for distributed applications. In *2017 New York Scientific Data Summit (NYSDS)*, pages 1–9.

European Organization For Nuclear Research and OpenAIRE (2013). Zenodo.

Foster, E. D. and Deardorff, A. (2017). Open science framework (osf). *Journal of the Medical Library Association : JMLA*, 105(2):203–206.

Guha, R. V., Brickley, D., and Macbeth, S. (2016). Schema.org: Evolution of structured data on the web. *Commun. ACM*, 59(2):44–51.

Heinrichs, B. and Politze, M. (2020). Moving towards a general metadata extraction solution for research data with state-of-the-art methods.

Herschel, M., Diestelkämper, R., and Ben Lahmar, H. (2017). A survey on provenance: What for? what form? what from? *The VLDB Journal*, 26.

Hu, R., Yan, Z., Ding, W., and Yang, L. T. (2020). A survey on data provenance in iot. *World Wide Web*, 23(2):1441–1463.

Interlandi, M., Ekmekji, A., Shah, K., Gulzar, M. A., Tetali, S. D., Kim, M., Millstein, T., and Condie, T. (2018). Adding data provenance support to apache spark. *The VLDB Journal*, 27(5):595–615.

Mufti, Z. and Elkhodr, M. (2018). *Data Provenance in the Internet of Things: Views and Challenges*.

Pérez, B., Rubio, J., and Sáenz-Adán, C. (2018). A systematic review of provenance systems. *Knowledge and Information Systems*, 57(3):495–543.

Politze, M., Claus, F., Brenger, B. D., Yazdi, M. A., Heinrichs, B., and Schwarz, A. (2020). How to manage it resources in research projects? towards a collaborative scientific integration environment. *European journal of higher education IT*, 1(2020/1):5.

Schmitz, D. and Politze, M. (2018). Forschungsdaten managen – bausteine für eine dezentrale, forschungsnahe unterstützung. *o-bib. Das offene Bibliotheksjournal / Herausgeber VDB*, 5(3):76–91.

Schwardmann, U. (2015). epic persistent identifiers for eresearch. In *Presentation at the joint DataCite-ePIC workshop Persistent Identifiers: Enabling Services for Data Intensive Research, Paris*, volume 21.

Smith, W., Moyer, T., and Munson, C. (2018). Curator: Provenance management for modern distributed systems. In *Proceedings of the 10th USENIX Conference on Theory and Practice of Provenance*, TaPP'18, page 5, USA. USENIX Association.

Talia, D., Thramboulidis, K., Lai, B. C., and Cao, J. (2013). Workflow systems for science: Concepts and tools. *ISRN Software Engineering*, 2013:404525.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., and Evelo, Chris T. ... Mons, B. (2016). The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3:160018.