Provenance as First Class Cloud Data

Kiran-Kumar Muniswamy-Reddy and Margo Seltzer
{kiran, margo}@eecs.harvard.edu
Harvard School of Engineering and Applied Sciences

Abstract

Digital provenance is meta-data that describes the ancestry or history of a digital object. Most work on provenance focuses on how provenance increases the value of data to consumers. However, provenance is also valuable to storage providers. For example, provenance can provide hints on access patterns, detect anomalous behavior, and provide enhanced user search capabilities. As the next generation storage providers, cloud vendors are in the unique position to capitalize on this opportunity to incorporate provenance as a fundamental storage system primitive. To date, cloud offerings have not yet done so. We provide motivation for providers to treat provenance as first class data in the cloud and based on our experience with provenance in a local storage system, suggest a set of requirements that make provenance feasible and attractive.

1 Introduction

Provenance is meta-data that describes the history of an object. In digital systems, this translates to a description of how the object was derived. Provenance is crucial in a number of areas such as scientific computation, security, regulatory compliance, and data archival. Scientists use provenance for experimental reproducibility or to determine what changed between two runs of an experiment. Provenance is useful for security as it can be used to verify how a virus spread through a system. The business community uses provenance to prove facts about information disclosure. Archivists maintain provenance meta-data to support document viability, renderability, understandability, authenticity, and identity in preservation contexts [15].

Provenance can be abstractly defined as a directed acyclic graph (DAG). The nodes in the DAG represent objects such as files, processes, tuples, data sets, etc. These nodes are further annotated with attributes. For example, a process node might be annotated with attributes such as the executable path, the environment, and the command line arguments. A file node is annotated with its name and version. The edge between two nodes indicates a dependency between the objects. For example, an edge from an object A to an object B indicates that B was derived from A. The provenance graph, by definition, is acyclic as the presence of cycles indicates that objects are their own ancestors.

Provenance increases the value of data on the cloud, as it does in other domains. For example, Amazon's "Public Data Sets on AWS" provides free storage for public data sets such as GenBank [2], US census data, and PubChem [16]. Provenance can help validate the processes that were used to generate these data sets and hence can help researchers decide if they want to use the data-sets. Alternately, hardware and software sometimes have bugs, and data released on the cloud may have been produced by such faulty hardware or software. For example, computations performed on cloud services, such as EC2 (Amazon's virtual machine service), make use of machines whose hardware, operating system, and library configurations are frequently updated. Provenance can help identify if a data set was tainted by faulty hardware or software.

While provenance is useful to storage users, it also provides valuable hints about the data to the cloud storage providers. For example, while a cloud store is making prefetching decisions, it can use the provenance graph to drive the prefetching decisions as all objects in the graph might be accessed together. Similarly, providers can let users set access control policies on their data via predicates on the provenance. Access policies based on provenance are succinct and less error prone. We discuss these use cases in detail in Section 2.

Most cloud storage services are not designed to conveniently store meta-data, let alone provenance. Hence, users currently have to store provenance in a separate service. As provenance is neither generated nor maintained as a core part of the storage service, storage providers cannot connect provenance to the data and thus cannot exploit the information inherent in provenance. Not only is this inconvenient to users as they have to develop additional tools to record provenance, but it is also a missed opportunity for cloud service providers. Providing strong support for provenance is a win-win situation for cloud providers; it reduces the development effort for cloud

users and provides information of high value to cloud providers.

The contributions of this paper are:

1. We demonstrate the utility of provenance to cloud storage providers.
2. We illustrate how current cloud meta-data support is insufficient for provenance.
3. We present a set of requirements that will make provenance accessible to users and useful to cloud providers.

The rest of the paper is organised as follows. In Section 2, we present use cases that illustrate how cloud providers can exploit provenance. In Section 3, we present background on some of the current cloud services and discuss how their interfaces are inadequate for provenance storage and management. In Section 4, we present the requirements that cloud storage providers have to satisfy in order to treat provenance as a first class citizen. We conclude in Section 5.

2 Why Cloud Store Providers Should Care About Provenance

Provenance provides information about an application and its structure (via the DAG) that can enable providers to enhance their service. In this section, we describe use cases that demonstrate how cloud storage providers can use provenance to more effectively manage their storage and augment the value of the services they provide. We also present our thoughts on the privacy concerns that users might have regarding storing provenance on the cloud.

2.1 Detect Application Anomalies

The pay-as-you-go business model of the cloud presents an interesting dynamic. Consider an application that fails to shut down appropriately, thereby consuming more resources than intended. On one hand, it might be in a cloud provider's best interest to allow such behavior, because they can charge for the unintended resource consumption. On the other hand, it is in the provider's best interest to earn the good will of customers, alerting them to anomalous behavior (especially behavior that might be costing them money).

Provenance can help providers in this endeavor. Provenance provides a detailed representation of the "normal" behavior or use of programs or objects. Changes in the structure of the provenance are a reasonably good indication of unusual behaviour. Prior work [7, 20] in security uses information similar to provenance to detect that a system is behaving in an anomalous fashion. These systems use runtime data to construct a model of "normalcy." Using these models, they detect applications behaving in unusual ways and take appropriate corrective actions.

Similarly, cloud stores can use provenance to build models of both applications and users. Just as credit card companies alert customers when their card usage patterns appear anomalous, cloud providers could similarly alert their customers of unusual behavior [1].

For example, data generated by a `video-encoder` application might be write-once, read-mostly. If, for some reason, a bug in the application is triggered that starts over-writing the same file over and over, the cloud could detect this anomalous behaviour and pro-actively warn the user.

2.2 Content-Based Search

As users store more shared data on the cloud, cloud providers will need to provide content-based search to help users find data of interest. The cloud search problem is synonymous to web search in many ways. Web search engines have a centralized location with large compute and storage power to index the web. Similarly, the cloud is centralized with a large amount of compute and storage power. One major difference between the cloud and the web is that cloud data lacks the explicit link structure present in the web; this link structure plays an important role in improving the efficacy of web search.

Provenance, however, forms its own graph and can play a similarly important role in enhancing data search in the cloud. Shah et. al. [18] showed that provenance can be used to improve desktop search. Shah's scheme computes an initial seed of search results using content-based search and then enhances the results using the provenance dependency links of the files in the result. Similarly, cloud stores can compute search results based on the content and then further refine the results using provenance that describes inter-file dependencies.

2.3 Provenance as Access Control

Currently, cloud stores provide access control via Access Control List (ACL). Since access control in every cloud offering varies in the details, we use Amazon's Simple Storage service (S3) [17] as an example for the rest of this section. In S3, users either specify explicit ACLs for each object they wish to share or make them publicly available to all Amazon Web Services (AWS) users or

[1] We acknowledge that building accurate models of application behaviour remains an open research area and that there can be false negatives. Even so, provenance can augment existing anomaly detection mechanisms or can provide data to explain why anomalies occur.

all internet users. Restricting access is tedious and error prone as it is easy for users to inadvertently forget to set permissions or to set them incorrectly. Another alternative is for users to group data into containers called buckets and set the appropriate access control for the buckets. However, such a re-organization of data is tedious and requires that users organize data logically in a way that matches their security requirements.

Often the process used to produce an object provides hints about appropriate access control for that object. For example, data derived from confidential data probably needs to remain confidential, while data derived by "blessed" aggregating processes can probably be made public. Hence, if cloud providers treat provenance as a first-class entity, they can allow users to specify access control policies as a function of the provenance. For example, a user can specify a policy that states all data derived from a file named "top-secret" should be available only to a specified set of users and that all files generated by an "aggregator" program can be accessible to all users. Compared to simple ACLs, this is easier to specify and manage due to its succinctness. This is also a finer-grained approach relative to more sophisticated ACL systems, such as ones based on file-name extensions. Provenance based access control can also be used to provide reasonable defaults for access controls.

Further, the succinctness of the provenance model makes it easier to change access policies. In the ACL model, a change in policy requires a careful review and modification of all the affected objects and users. In contrast, in the provenance model, policy changes only require reviewing and updating the relevant high level policies. Such a change is much easier to reason about and review.

2.4 Object Clustering/Pre-fetching

A number of research projects have explored the issue of identifying clusters of related objects [9, 8, 5] for applications ranging from hoarding objects to pre-fetching objects. These projects were conducted in the context of local file systems and in the context of mobile computers operating while disconnected. One can imagine adapting these algorithms for the cloud to derive similar benefits.

Provenance can play a central role in adapting the algorithms. Dependency links readily provide hints on how data are related to each other. For example, the dependency links might suggest that a particular set of objects in a graph are regularly accessed sequentially and hence it might be beneficial to prefetch them as soon as the first one is requested. Further, the decisions used to perform object prefetching, can in turn drive object placement decisions. For example, since objects are prefetched together, it might make sense to ensure that the objects reside on a disjoint set of hardware (nodes, routers, etc.). Furthermore, since many clients store data on the cloud, the cloud can build more robust models for clustering, prefetching, and object placement by aggregating provenance graphs/access patterns for similar data across multiple users.

Finally, cloud providers could use the provenance to go one step further. Since provenance identifies the process and the inputs used to generate data and their access patterns, the provider can make explicit tradeoffs between actually storing the data or re-generating it on demand [1].

2.5 Discussion: Privacy Issues

One can argue that some of the use cases violate user or data privacy. Note, however, that using provenance is no different from the email service scenario. Sites like `www.gmail.com` mine the text of email messages to show relevant advertisements to users. In return users get free email service, providing good search capabilities. Similarly, we believe that it is ethical for cloud storage providers to use provenance to provide improved service to users as long as cloud storage providers do not exploit provenance in a malicious manner.

3 Current Cloud Storage Services

Having described how cloud store providers can utilize provenance, we now describe some of the current cloud services, the interfaces they provide, and how the interfaces they provide are inadequate for storing provenance.

3.1 Simple Storage Service (S3)

S3 is Amazon's storage service [17]. It is an object store capable of storing objects whose sizes range from 1 byte to 5GB. Each object is identified by a unique URI. Clients access S3 objects using either SOAP- or REST-based APIs. Users use the PUT operation to store an object on S3, overwriting an object if it already exists. With each object, clients can store up to 2KB of meta-data, represented as <name,value> pairs. The meta-data is also specified as a part of the PUT operation. The GET operation retrieves both the data and the meta-data. The HEAD operation retrieves only the meta-data part of an object. There are two significant limitations to storing provenance in the meta-data of an object on S3. First, 2KB is insufficient in many cases to store the entire provenance. Second, one cannot query by the meta-data. This is particularly debilitating as it prevents users from issuing queries on provenance.

Users who want to record provenance of data that they store on S3 use Amazon's SimpleDB [19] service to store

provenance [4]. The SimpleDB service provides index and query functionality necessary for provenance. SimpleDB's data model is semi-structured, i.e., it consists of a set of rows (called items), with each row having a unique itemid and each item having a set of attribute-value pairs. SimpleDB provides a SELECT query interface and a custom QUERY interface that users can use to retrieve provenance.

3.2 Microsoft Azure Blob

Microsoft Azure *Blob* is a service similar to S3. While the basic operations are the same in both services, they differ in the details. For example, Blob imposes an 8KB limit on the meta-data. S3 allows users to record objects as large as 5GB in one PUT, while Blob limits a single PUT to 64MB. To write larger objects in Blob, clients write blocks of data using a PutBlock operation and then issue a PutBlockList command that assembles an object from its blocks and makes the object visible. Similar to S3, Blob does not allow users to query the meta-data recorded with the data. The only option users have to lookup meta-data is to download all the meta-data and examine it.

As in the case of Amazon S3, users can use the Microsoft Azure *Table* service to store and retrieve provenance for objects in the Blob. The Table service is mostly similar to SimpleDB, with a major difference being that Azure supports a LINQ [10] query interface in contrast to SimpleDB's query interface.

3.3 Nirvanix Internet Media File System (IMFS)

IMFS [14] is a distributed clustered file system accessible over the Internet. As its name suggests, it is optimized for storing media files. Like S3 and Blob, it is meant to deal with large objects that are written rarely and read often. It provides SOAP- and REST-based APIs. Unlike the object interfaces of S3 and Blob, IMFS provides a file system interface. IMFS is more advanced in its meta-data management. It allows users to annotate objects with <name,value> pair meta-data and tags. Tags are strings that are set on objects. Further, IMFS allows users to lookup objects based on meta-data or tags. Objects matching meta-data can be looked up using the *SearchMetadata* method. The call takes a meta-data search key and a value and looks for objects that have matching meta-data on them. Similarly, objects matching a tag can be looked up using the *SearchTags* method. The method takes a search string to look for in the tags. Hence users can record provenance as meta-data attribute/value pairs, using the *SetMetadata* call. They can perform rudimentary provenance queries using the SearchMetadata call[2]. However, there is no support for complex queries, and in particular, no support for queries on paths (through a provenance DAG), which are fundamental to many provenance applications [6].

3.4 Summary

None of the current services provide explicit support for provenance. At best, users can treat provenance as meta-data, storing it separately in a database/table-like service. As a result, cloud stores cannot take advantage of the provenance.

4 Provenance Requirements

We now discuss some of the requirements that cloud storage providers must satisfy in order to make provenance a first class citizen.

Co-ordination Between Storage and Compute Facilities. Cloud vendors frequently provide both storage and compute facilities. For example, Amazon offers the EC2 virtual machine service that provides users with on-demand compute power. EC2 users have to use Amazon's storage services to persist their data. In scenarios such as this, the cloud provider, since it provides both storage and compute facilities, can provide virtual machines that can infer the provenance of the data being generated and transmit the provenance to the storage service along with the data. One approach to achieve this could be to provide virtual machine instances that are installed with operating system kernels such as Provenance-Aware Storage Systems [12] that automatically keep track of provenance.

Allow Customers to Record Provenance of their Objects. Customers might generate data locally on their systems and store the data on the cloud. Clearly, here the customer or the customer's local system must be responsible for tracking provenance and providing it to the cloud for storage. Hence the cloud should provide interfaces for customers to record provenance computations performed locally. If provenance is simply treated as data, most current cloud services meet this requirement. In this scenario, the cloud is a low level provenance substrate and the users are higher-level processes that have more detailed provenance. Users layer their provenance system on top of the cloud. In prior work, we explored the issues in layering provenance systems [11].

[2]Their documentation does not specify the limits on the size of the meta-data and the number of <key,value> pairs that can be stored with one object.

Provenance Data Consistency. In order to use provenance to manage the system, cloud providers should ensure that provenance is consistent with the data it describes. As the cloud is inherently a distributed system, if the provenance and the data are recorded using separate methods, there can be situations in which provenance becomes inconsistent with the data. For example, a system might crash after updating the data but before updating its provenance. The inconsistency can mislead both users and providers that want to use the data.

Cloud stores should provide interfaces that store provenance and data atomically, thus ensuring consistency between provenance and data. The current S3 and Blob interfaces already satisfy this requirement. However, as we discussed earlier, since their meta-data is not searchable, users have to store provenance in a database service such as SimpleDB or Table. This introduces the provenance-data consistency problem. In prior work, we discuss protocols for storing provenance in the database service such that the provenance is consistent with data [13]. IMFS, however, provides two separate methods for storing data and for storing meta-data, and hence does not satisfy this requirement.

Long-Term Persistence. Provenance has to be retained beyond the lifetime of the object it describes. The provenance of an object might connect objects that are otherwise unrelated. Hence removing the provenance when an object is deleted can result in the provenance chain being disconnected. If the object had no descendants, then the cloud store can choose to remove its provenance, since it would no longer be accessible via any provenance chain.

Expose Provenance for External Use. Providing the right interface for storing provenance allows users to store provenance and for the cloud to utilize it. However, provenance is primarily of interest to the users who want to use some data on the cloud. For example, provenance must be accessible to users who want to verify properties of their data or simply be aware of its lineage. If a system stores provenance, but that provenance is not readily accessible, the provenance is of questionable value. Thus the cloud must support efficient query and indexing of provenance.

Provenance Security. Provenance can potentially contain sensitive information. For example, consider a paper review: the data (contents of the review) must clearly be accessible to the authors, but the provenance (reviewers' identities) must not be. Hence the cloud storage provider should ensure appropriate security for provenance itself. Provenance security is, however, an open problem as the provenance and the data it describes do not necessarily share the same access control [3].

5 Conclusions

In this paper, we argue that cloud stores should treat provenance as a first class citizen. By treating provenance as a first-class citizen, both the cloud store and the users will benefit. The users will benefit as it reduces the extra development effort needed to store provenance. In turn, cloud providers can take advantage of the rich information inherent in provenance for various applications ranging from security enforcement, to search, and to performance improvement.

Acknowledgments: We thank Greg Morrisett and Stephen Chong for their feedback on early drafts of the paper. We thank the LADIS reviewers for the valuable feedback they provided. This work was partially made possible thanks to NSF grant CNS-0614784.

References

[1] I. Adams, D. D. E. Long, E. L. Miller, S. Pasupathy, and M. W. Storer. Maximizing efficiency by trading storage for computation. In *Proceedings of the Workshop on Hot Topics in Cloud Computing (HotCloud 09)*, 2009.

[2] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. Genbank. *Nucleic Acids Research*, 36 (Database Issue), January 2008.

[3] U. Braun, A. Shinnar, and M. Seltzer. Securing Provenance. In *Proceedings of HotSec 2008*, July 2008.

[4] C. Dagdigian. Plenery Keynote: Bio.IT World. `http://blog.bioteam.net/wp-content/uploads/2009/04/bioitworld-2009-keynote-cdagdigian.pdf`.

[5] J. Griffioen and R. Appleton. Reducing file system latency using a predictive approach. In *USTC'94: Proceedings of the USENIX Summer 1994 Technical Conference on USENIX Summer 1994 Technical Conference*, pages 13–13, Berkeley, CA, USA, 1994. USENIX Association.

[6] D. A. Holland, U. Braun, D. Maclean, K.-K. Muniswamy-Reddy, and M. I. Seltzer. A Data Model and Query Language Suitable for Provenance. In *Proceedings of the 2008 International Provenance and Annotation Workshop (IPAW)*, June 2008.

[7] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen. Enriching intrusion alerts through multi-host causality. In *the 12th Annual Network and Distributed System Security Symposium*, 2005.

[8] T. M. Kroeger and D. D. E. Long. Predicting file system actions from prior events. In *ATEC '96: Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*, pages 26–26, Berkeley, CA, USA, 1996. USENIX Association.

[9] G. H. Kuenning. The design of the seer predictive caching system. In *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 37–43, 1994.

[10] The LINQ project. `http://msdn.microsoft.com/en-us/vcsharp/aa904594.aspx`.

[11] K.-K. Muniswamy-Reddy, U. Braun, D. A. Holland, P. Macko, D. Maclean, D. Margo, M. Seltzer, and R. Smogor. Layering in provenance systems. In *Proceedings of the 2009 USENIX Annual Technical Conference*, June 2009.

[12] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer. Provenance-aware storage systems. In *Proceedings of the 2006 USENIX Annual Technical Conference*, June 2006.

[13] K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer. Making a cloud provenance-aware. In *1st Workshop on the Theory and Practice of Provenance*, 2009.

[14] Nirvanix internet media file system. `http://developer.nirvanix.com/sitefiles/1000/API.html`.

[15] Data Dictionary for Preservation Metadata. `http://www.oclc.org/research/projects/pmwg/premis-final.pdf`, May 2005.

[16] Pubchem. `http://pubchem.ncbi.nlm.nih.gov/`.

[17] Amazon Simple Storage Service (Amazon S3). `http://aws.amazon.com/s3`.

[18] S. Shah, C. A. N. Soules, G. R. Ganger, and B. D. Noble. Using provenance to aid in personal file search. In *Proceedings of the USENIX Annual Technical Conference*, pages 1–14, Berkeley, CA, USA, 2007. USENIX Association.

[19] Amazon SimpleDB. `http://aws.amazon.com/simpledb`.

[20] A. Somayaji and S. Forrest. Automated Response Using System-Call Delays. In *USENIX Security Symposium*, Berkeley, CA, 2000.