Open access • Proceedings Article • DOI:10.1109/CLOUDCOM.2011.66

How to Track Your Data: The Case for Cloud Computing Provenance — Source link

Olive Qing Zhang, Markus Kirchberg, Ryan K. L. Ko, Bu-Sung Lee

Institutions: Hewlett-Packard

Published on: 29 Nov 2011 - IEEE International Conference on Cloud Computing Technology and Science

Topics: Cloud computing security and Cloud computing

How To Track Your Data: The Case for Cloud Computing Provenance

Olive Qing Zhang, Markus Kirchberg, Ryan K L Ko, Bu Sung Lee

HP Laboratories
HPL-2012-11

Keyword(s):

cloud computing; data provenance; accountability; transparency;

Abstract:

Provenance, a meta-data describing the derivation history of data, is crucial for the uptake of cloud computing to enhance reliability, credibility, accountability, transparency, and confidentiality of digital objects in a cloud. In this paper, we survey current mechanisms that support provenance for cloud computing, we classify provenance according to its granularities encapsulating the various sets of provenance data for different use cases, and we summarize the challenges and requirements for collecting provenance in a cloud, based on which we show the gap between current approaches to requirements. Additionally, we propose our approach, DataPROVE, that aims to effectively and efficiently satisfy those challenges and requirements in cloud provenance, and to provide a provenance supplemented cloud for better integrity and safety of customers' data.

External Posting Date: January 21, 2012 [Fulltext] Approved for External Publication
Internal Posting Date: January 21, 2012 [Fulltext]

How to Track Your Data: The Case for Cloud Computing Provenance

Olive Qing Zhang, Markus Kirchberg, Ryan K L Ko, and Bu Sung Lee
Cloud & Security Lab, Hewlett-Packard Laboratories, Singapore
Email: Y060098@e.ntu.edu.sg, {Markus.Kirchberg | Ryan.Ko | Francis.Lee}@hp.com

***Abstract*—Provenance, a meta-data describing the derivation history of data, is crucial for the uptake of cloud computing to enhance reliability, credibility, accountability, transparency, and confidentiality of digital objects in a cloud. In this paper, we survey current mechanisms that support provenance for cloud computing, we classify provenance according to its granularities encapsulating the various sets of provenance data for different use cases, and we summarize the challenges and requirements for collecting provenance in a cloud, based on which we show the gap between current approaches to requirements. Additionally, we propose our approach, DataPROVE, that aims to effectively and efficiently satisfy those challenges and requirements in cloud provenance, and to provide a provenance supplemented cloud for better integrity and safety of customers' data.**

I. Introduction

Cloud computing is defined in [1] as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Cloud computing can be distinguished by its appearance of infinite computing resources that can be upgraded or downgraded freely on demand, and its pay-per-use billing feature as utility computing.

Provenance, also referred as lineage and pedigree, is generally defined as the information that helps determine the derivation history of a data product, starting from its original resources [2]. The provenance of a data product mainly consists of two parts: the ancestral data products from which the data product is derived, and the process of transformation of these ancestors that derive the data product.

In past decades, provenance has been supplemented to many different domains and applications [3]–[5], where its effectiveness is widely appreciated in a number of important functions such as explanation, verification, and re-computation. Provenance can be used for scientists to examine the sources and evolution of data products of interest, to determine the differences between two runs of an experiment, and to reproduce the result of an experiment. Provenance can be used for security by tracing how a virus spreads out in a system. Provenance can be used for business community to identify information disclosure and to detect insider trading. Furthermore, provenance can be used for developers and administrators to identify the source of system failures and to debug complex programs consisting of multiple modules.

Tremendous amount of research has been done in supplementing provenance from various layers of abstraction, ranging from application level [6]–[8], work-flow level [9]–[12], and system level [13], [14]. However, considering only a single layer of provenance is not sufficient in many practical scenarios; one such example was at the core of the first provenance challenge [15], which required integration of provenance to answer questions that necessitate an integrated view of provenance. The PASSv2 system [16] facilitates the integration of provenance across multiple levels of abstraction, including a work-flow engine, a Web browser, and an initial run-time Python provenance tracking wrapper. Layering these components atop a provenance-aware network storage, PASSv2 is able to address the challenges proposed in [15].

Provenance is particularly crucial for cloud computing, reasons including:

- Data can be shared widely and anonymously in the cloud, provenance is required to verify the authenticity or identity of data [17].
- Some scenarios in cloud computing have clear requirements for provenance of data, such as eScience [18] and healthcare [19], where assurance in the quality of repeatability of results is essential.
- Clouds have their own application for provenance, i.e., detection of the origins of faults and security violations [20].
- Provenance is also useful to cloud storage providers; either increasing the value of data or providing valuable hints about data. Use cases include anomalies detection, content-based search, provenance-based access control policies, and object clustering/pre-fetching [21].
- In [22], data confidentiality/auditability was listed as the third obstacle for better adoption of cloud computing. Cloud users face security threats both from outside and inside the cloud, possibly even from cloud service providers. In a 2010 survey by Fujitsu Research Institute [23] on potential cloud customers, it was found that 88% of potential cloud consumers are worried about *who has access to their data*, and demanded more awareness of what goes on in the back-end physical server (i.e., virtual and physical machines). If provenance can be provided in cloud, users are given more control over their data, in particular being able to verify that nothing has been done wrong, or, alternatively, detect what went

wrong w.r.t. the data under their control.

- There are many legal issues, regulations, policies, and governance rules connected with cloud computing, either formally documented or to be constructed; Mowbray discussed many such problems in [24]. Many countries and regions have their privacy laws and data protection laws, such that legal disputes may arise from geopolitical and jurisdictional issues in using geographically dispersed cloud computing services. For instance, some organizations are wary about the possibly that cloud computing services are processing or storing their data outside of the country of origin which may enable foreign governments to access their data.By recording physical locations of storage and the occurrence of cross-border data transfers, cloud provenance can help to trace whether such laws and regulatory regimes hold, and notify customers when their confidential data is or has been accessed.

In this paper, we survey current mechanisms that support provenance for cloud computing, we classify provenance according to its granularities encapsulating the various sets of provenance data for different use cases, and we summarize the challenges and requirements for collecting provenance in a cloud, based on which we show the gap between current approaches to requirements. Additionally, we propose our approach, DataPROVE, that aims to effectively and efficiently satisfy those challenges and requirements in cloud provenance, and to provide a provenance supplemented cloud for better integrity and safety of customers' data.

II. Related Work

Previous researches on provenance have mainly focused on database systems [25]–[27], file systems [14], [28], work-flow systems [9]–[12], and operating systems [16]. With the rapid adoption of cloud computing, there has been an increase in research on provenance in cloud computing. In [20], [21], [29], incorporating provenance as first class cloud data is motivated, and the constraints and challenges in doing so are discussed. In [17], [30], Muniswamy-Reddy *et al.* proposed four properties that make provenance system truly useful, and implemented three protocols for maintaining provenance in current cloud stores. They also attempted to collect provenance using Xen hypervisor [31], which proceeds one step ahead in incorporating provenance to cloud computing.

A. PASS and its Provenance Collection via Xen Hypervisor

Provenance-aware Storage System (PASS) is a pioneer system that treats provenance as first-class object, via automatic provenance collection and maintenance. Improved from operating at only one level of abstraction, the second PASS prototype presented in [16] facilitates the integration of provenance across multiple levels of abstraction to better support users' needs to reason about their data and processes, and to answer questions that require an integrated view of the provenance. The prototype consists of a central Data Provenance API (DPAPI) and seven main components, namely *libpass*, *interceptor*, *observer*, *analyzer*, *distributor*, *Lasagna*, and *Waldo*. The DPAPI allows transfer of provenance both among system components and across layers, which is exported to user-level by library *libpass*. Provenance-aware applications can be developed by augmenting code to collect application-specific provenance, and issuing DPAPI calls to *libpass*. The *interceptor* intercepts system calls and passes information to the *observer* to generate provenance records. The *analyzer* processes the stream of provenance records and eliminates duplicates and cycles. The *distributor* caches provenance for non-persistent objects until they need to be materialized on disk. *Lasagna* is a provenance-aware file system that stores provenance records along with data these records describe; provenance records are written internally to a log whose format ensures consistency between the provenance and data. The user-level daemon *Waldo* reads provenance records from the log, stores them in a database, indexes them, and accesses the database for querying.

To facilitate cloud computing, Macko *et al.* extend the second PASS prototype and modify the Xen hypervisor to collect provenance from running guest kernels; details are given in [31]. The approach places an *interceptor* on a DomU (i.e., a guest virtual machine) to intercept system calls in Xen's syscall_enter mechanism, and stores provenance records in a ring buffer. Concurrently, a user space daemon runs in a privileged domain (i.e., Dom0 or a special "provenance-processing domain") to periodically consume records from the ring buffer, and perform other tasks related to the *analyzer* and *Waldo* similarly as in the second PASS prototype. Authors also already identify several pitfalls of their proposed approach; these include the requirement of a large ring buffer to hold all records that have not been consumed, increased demands on processor caches and traffic to the CPU interconnect by buffer operations, the multiple-to-one correspondence between path strings and the object, and loss of system call records due to hardware failure or a software bug in Xen or the Dom0 kernel. Besides these pitfalls, there are also gaps with respect to providing complete provenance within a cloud;gaps include:

- Tracking of provenance up to the physical hardware layer in order to capture the mapping between virtual and physical resources (e.g., for facilitating forensic investigations) and control flow of data transfers.
- Without presence of provenance-aware network protocols, provenance chains will break if provenanced data is transferred from one DomU to another DomU inside a cloud; as such, the provenance of data in the cloud would be isolated to the respective DomU, making cloud-wise provenance query difficult.
- Security mechanisms (including access control policies of different confidentiality levels) for provenance should be mandated to assure integrity and confidentially of provenance information such that provenance data cannot be forged or tampered.

B. PASS Provenance Recording Protocols

In [17], PASS is used as the provenance collection substrate and extended to use Amazon Web Services [32] as the

storage back-end. Specifically, PASS monitors system calls of a client (i.e., compute node), generates provenance, and sends both provenance and data to PA-S3fs, which is a user-level provenance-aware file system interface for Amazon's S3 storage service by exporting DPAPI. PA-S3fs caches data in a local temporary directory and provenance in memory, and sends them to the cloud using one of three protocols (i.e., P1, P2, or P3) as appropriate. Properties of each of these three protocols are as follows:

1) P1, Standalone Cloud Store: P1's storage scheme maps each file to a primary S3 object and stores the object's provenance as another S3 object. The primary S3 object contains the data, and the provenance S3 object (identifiable through a *uuid*) contains the primary S3 object's provenance plus the name of the primary S3 object. The version number and the *uuid* is recorded in the meta-data of the primary S3 object, such that the data and its provenance link together. For objects not persistent from the kernel perspective, only the provenance object is recorded with no primary object. Using P1, on a file close or flush, the provenance of a file is placed into an S3 object, after which the data object with its meta-data attributes is encapsulated. Before sending the provenance and the data, the ancestors of the object are identified and, if unrecorded, made persistent to ensure multi-object causal ordering (discussed in Section IV-B).

P1 does not support data-coupling, but decoupling of provenance from data can be detected using version numbers stored in both the provenance object and the primary object's meta-data. The implementation for ensuring multi-object causal ordering can suffer from high latency. Also, efficient querying of provenance is not supported as the P1 protocol requires a look-up of the primary object and retrieval of its provenance. Thus, resulting in expensive iterations over all objects in the repository if the exact object, whose provenance is of interest, is unknown.

2) P2, Cloud Store with a Cloud Database: P2's storage scheme stores each file as a primary S3 object and its provenance in SimpleDB [33]. The provenance of a version of an object is stored as an item in SimpleDB; referencing is achieved by assigning a *uuid* to each object at creation time and appending the version number during object versioning. Provenance values larger than 1KB SimpleDB limit are stored as separate S3 objects and referenced from items in SimpleDB. The version number and the *uuid* are recorded in the meta-data of the primary S3 object as with P1. On a file close, the provenance information cached in memory is extracted, converted to attribute-value pairs, and grouped by file versions. Using P2, any value larger than 1KB is stored as S3 object, and referenced by pointer in the respective attribute-value pair. Provenance is stored in SimpleDB as items by issuing batch-update calls. Finally the data object with meta-data attributes are placed into an S3 object.

P2 is an improvement over P1 in that it provides efficient querying through indexed provenance in SimpleDB. Same as with P1, P2 also exhibits high latency when ensuring multi-object causal ordering.

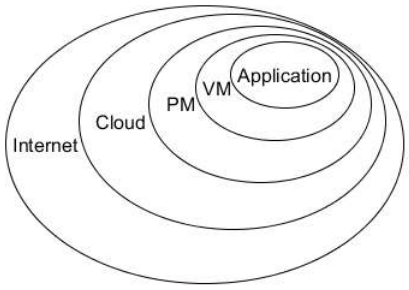


Fig. 1. Conceptual View of Provenance Granularities.

3) P3, Cloud Store with Cloud Database and Messaging Service: P3 uses the same S3/SimpleDB storage scheme as P2, but differs from P2 in its use of a cloud messaging service (SQS) and transactions to ensure provenance data-coupling. Each client has an SQS queue, which is used as a write-ahead log (WAL) and a commit daemon, reading the log records and assembling all the records belonging to a transaction. Once the SQS queue holds all the records of a particular transaction, the daemon pushes data in the records to S3 and corresponding provenance information to SimpleDB. If the client crashes before it can log all the packets of a transaction to the WAL queue, the commit daemon ignores these records. P3 operates in two phases: log and commit. On a file close or flush, the log phase begins to store a temporary data copy in S3, extract the provenance of the object, and arrange provenance into messages in the WAL queue in the designed format. In the commit phase, the commit daemon assembles packets belonging to transactions. Once all packets of a transaction are received, it stores provenance in a similar way as with P2, copies the temporary S3 object to a permanent one, deletes the temporary copy, and deletes all messages related to the transaction.

P3 is an improvement over P2 as it also satisfies provenance data-coupling through the use of an SQS queue.

III. Granularities of Provenance in the Cloud

Provenance in a cloud shares similar definition as in a local system (i.e., a single machine), but its context is much broader than that of a local system due to the infrastructure of the cloud. In our approach, we focus on data-centric provenance across system and application layers in a cloud [34]. With this approach, data-centric provenance can be divided into five granularities based on the type of resources utilized, i.e., application, virtual machine, physical machine, cloud, and Internet. The conceptual view and component view of these granularities are given in Fig. 1 and Fig. 2.

1) Provenance of Application: This granularity summarizes all provenance data collected from user applications across Software-as-a-Service (SaaS) and Platform-as-a-Services (PaaS), such as word processor, web-based emails, and Google App Engine, where focus is given to data processing and file manipulation.

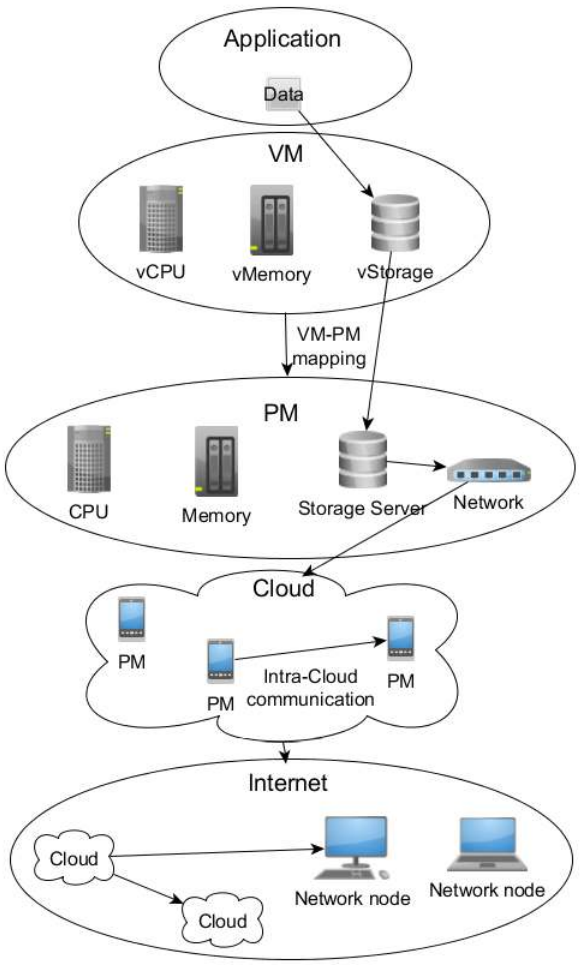


Fig. 2. Component View of Provenance Granularities.

2) Provenance of Virtual Machine (VM): This granularity summarizes all provenance data specific to a VM where provenance is collected, such as virtual resource status, kernel version, operating system, modules loaded, library configurations, the amount of main memory available, the memory allocated to the address space, file path of an object on the VM etc.
3) Provenance of Physical Machine (PM): This granularity summarizes all provenance data specific to a PM where provenance is collected, such as the physical resources status, the physical address of memory/block devices/files, and the mapping of VMs to PMs.
4) Provenance of Cloud: This granularity summarizes all provenance data related to a cloud, such as customer identification information and intra-cloud networking which is communication between different PMs within a cloud, transfers of data across VMs and PMs located across different geographies in a cloud, the control flow of transfers of data (e.g. compliance of data transfers according to governance regulations such as the Sarbanes-Oxley (SOX) Act and Payment Card Industry (PCI) standards).
5) Provenance of the Internet: This granularity includes all provenance data related to data sharing on the Internet, including data transfer between different clouds, and from a cloud to any network node that does not belong to a cloud, which gives the broadest granularity.

To demonstrate these granularities, we use two simple examples describing the provenance and categorize them into different granularities (depicted in Table I). In Example 1, a researcher uses a public data set shared in the cloud as an input to his/her program, and stores the result generated in the cloud. Whereas in Example 2, a group of people share some confidential file in the cloud.

IV. Challenges and Requirements of Supplementing Provenance for Cloud Computing

A. Challenges

Many research works have discussed the various challenges in collecting provenance for cloud computing. In [20], Abbadi and Lyle discussed the difficulty in providing cloud provenance through linking together log and audit data collected from multiple resources, considering the dynamic and complex nature of cloud infrastructure, and proposed that a purpose-built provenance system should be designed for cloud infrastructure.

Sakka *et al.* discussed provenance constraints covering legal, business, and technical aspects based on a bank record case study in [29], where they divided challenges for provenance into known provenance challenges and new challenges arising from cloud context. Known provenance challenges include object identification between different clouds, provenance data-coupling and persistence, provenance reliability and confidentiality, as well as provenance interoperability; whereas challenges in a cloud context include extensibility, availability and scalability, which is also described in detail in [17].

Besides aforementioned challenges, we further identify some challenges involved in supplementing provenance for cloud computing covering different perspectives:

- Challenges introduced by virtualisation: Elasticity as a promise of cloud computing is empowered by virtualisation. In our earlier work [34], we explained the need of tracking virtual-to-physical mapping and vice versa, in order to provide transparency for the customers of the linkage between virtual and physical operating systems, virtual locations and physical static server locations, and to demonstrate how files are written into both virtual and physical memory addresses.
- Challenges introduced by dynamic and heterogeneous nature of clouds: Clouds are diverse in structure, configuration, availability, and service providers. Beside interoperability, provenance and provenance collection mechanism in clouds should be independent of diversities to suit the needs of different customers. For example, many different operating systems are available for customers to install on VMs in a cloud. It is infeasible to generate a version

TABLE I
PROVENANCE OF VARIOUS GRANULARITIES

Granularity	Example 1	Example 2
Application	Validate the processes for generating the data set and the result that helps the researcher to decide if the data set is to be used and to assure the quality of the result.	Understand how the file is created and where the information comes from.
VM	Obtain environmental information about VMs to ensure the reproducibility of the result; debug experimental results to determine the changes between invocations.	Check whether the file has been accessed by some malicious party using the same VM.
PM	Help identify if the input/output was tainted by faulty hardware; detect incidents on the same PM that corrupts the output.	Check whether the file has been accessed by some malicious party accessing the same PM.
Cloud	Verify the public data set used in the program to make sure that the input is reliable; discover how far faulty data has propagated within the cloud.	Detect whether a cloud administrator with full access rights tampering the file from anywhere inside the cloud; attest the integrity and enforcement of cloud resource policies; detect data transfer across geographic boundaries.
Internet	Discover the entire file transmission route; understand the discrepancy caused by inter-cloud manipulation.	Discover all executions done to the file, e.g., who/where on Internet has read/modified/downloaded the file.

of provenance collection for each operating system, nor enforce a single operating system for provenance. In [34], [35], it is also pointed out that a snapshot of a running, or "live" system such as the VMs turned on within a cloud can be only reproduced up to its specific instance and cannot be reproduced in a later time-frame. This demands cloud provenance of complex real-time characteristics. Time tagged provenance might address this issue, but time synchronization in a cloud poses another set of challenges in itself.

- Challenges introduced in fault tolerance and resilience of clouds: It is common that some PMs in a cloud go down because of hardware failure. Under such condition, transparent live migration of VMs residing on faulty PMs will take place transparently and unbeknown to customers. Provenance collection mechanisms should be able to capture necessary information involved in live migration, to assist in system debugging and forensic investigation. On the other hand, provenance data should not be lost or corrupted during hardware failure.
- Challenges introduced in collecting provenance of different levels of granularity:
  - Application: A well designed GUI/API is required to permit user annotation and application-specific provenance, which might be unknown to automatic provenance collection and provide more semantic knowledge of data described by provenance.
  - VM: Performance is of great concern in this granularity, because provenance collection will affect negatively the performance of a VM, especially if the VM is hosting I/O intensive applications.
  - PM: Challenges in virtual-to-physical mapping, as discussed above under virtualisation.
  - Cloud & Internet: Provenance-aware network protocols should be designed for intra-cloud, inter-cloud, and cloud-to-Internet data transfer, such that data and its provenance can be transferred together for consistency.

B. Requirements and Properties of Cloud Provenance Systems

We now discuss the core requirements and properties of cloud provenance systems that address common challenges listed, and make provenance in cloud truly useful. Table II summarizes the provenance collection mechanisms and systems discussed in Section II in terms of requirements and properties defined below.

- Coordination between storage and compute facilities: Provenance, like typical digital data in a cloud, is generated and processed by compute facilities, and persisted on storage facilities [21]. Coordination is supported by cloud service providers since they usually provide both facilities (e.g., Amazon provides EC2 compute facilities and S3 storage services).
- Interface/API that allows customers to record provenance of their objects: A complete picture of provenance needs every piece of jigsaw to be provenance-aware, ranging from application, local system, cloud system, to network transmission. If data originated from non-provenanced sources is to be stored on the cloud, a solution to incomplete provenance is to allow verified/signed manual input of provenance for the data, with the help of a well designed interface and/or API.
- Security elements that need to be provided for reliable provenance:
  - Integrity: The assurance that provenance is not forged or tampered [29].
  - Auditability: An auditor can check the integrity and the correctness of provenance information [29], though how to prohibit or detect suspicious user annotation and false provenance fabricated by malware is still an open question.
  - Confidentially: Provenance may contain sensitive information of the data it describes, or it may be sensitive information by itself [21], [29]. Encryption methods and access control policies for provenance are a necessity to prevent information leakage from provenance. It is difficult to ensure confidentially

TABLE II
SPECIFICATION OF PROVENANCE SYSTEMS

Provenance Systems	Provenance Collection		Provenance Security	Provenance Content				
	Coordination	Interface/API		Consistency	Atomicity	Causal Ordering	Persistence	Efficient Query
Provenance collection in Xen [31]	√	√	×	–	–	–	–	–
P1 [17]	–	–	×	×	×	√	√	×
P2 [17]	–	–	×	×	×	√	√	√
P3 [17]	–	–	×	√	√	√	√	√
DataPROVE	√	√	√	√	√	√	√	√

when inside intruders such as privileged administrators and cloud service providers are involved.

- Provenance data consistency: Provenance information must be consistent with data it describes [21]. Inconsistency in provenance and its data can mislead both customers and service providers.
- Atomicity: Provenance must be recorded atomically with the data it describes [30]. Atomicity pertains to provenance storage, whereas consistency pertains to provenance and data retrieval. Atomicity and consistency together assures that provenance accurately and completely describes the data, i.e., provenance data-coupling.
- Causal ordering: A provenance system must ensure that an object's ancestors and their provenance are persistent before making the object itself persistent [17], which prevents dangling pointers from appearing in provenance recorded as a directed acyclic graph (DAG).
- Data independent persistence, also referred to as long-term persistence: A provenance system retains an object's provenance even after the object is removed [17]. Although an object is removed, its provenance must still be present in the provenance DAG as some other objects' ancestor; deleting the object's provenance will make the DAG disconnected. An object's provenance can be removed only if it has no descendants.
- Efficient query: The primary use of provenance data is for users to check lineage properties of a corresponding object of interest, through external queries [17]. Considering the graph structure of provenance and large size of a cloud and objects stored in it, efficiency of querying affects directly the value of provenance.

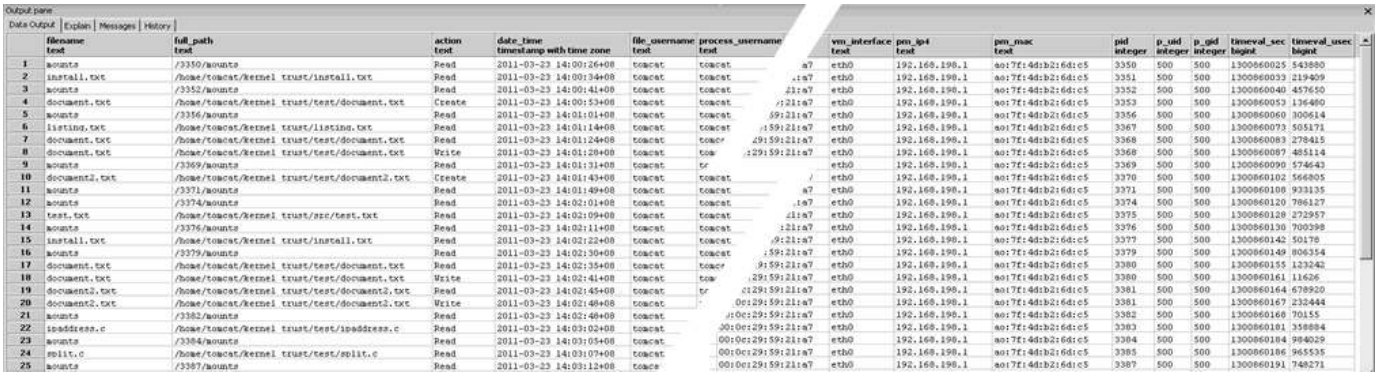
In their earlier works, Muniswamy-Reddy *et al.* introduced several of the requirements and properties discussed above. They also proposed approaches to either collect provenance in a local machine, and store data and its provenance in the cloud through different recording protocols, or collect provenance via a hypervisor that manages the VMs in a cloud. However, their work reaches only the VM granularity that we defined in Section III. As we extend our thoughts further down the road, we realize that there is always non-provenanced data on the Internet, unless every piece of processing tool, either standalone or online, is provenance-aware. As this is unlikely to be the case, we strongly believe that there should also be mechanisms for a provenance-system to permit explicit input of provenance information (using appropriate verification protocols). For such circumstances, a universal interface/API can be useful enabling users to manually input provenance for their non-provenanced data, as well as for developers to layer their provenance-aware programs atop cloud provenance systems. Such a interface will also permit a third-party, provenance-aware applications to send provenance information to cloud provenance systems. Considering this, we motivate interface/API as an additional requirements for cloud provenance systems.

V. THE DATAPROVE APPROACH

Our current work addresses issues identified in the Data Layer of the TrustCloud framework [34]. TrustCloud attempts to increase trust in cloud service providers through heightened data transparency and accountability. This framework encapsulates accountability in the Cloud via five layers of granularity:

- System Layer: Monitoring of containers of data, e.g., files and databases, in the Cloud.
- Data Layer: Monitoring of data evolutions and creations in the Cloud; provenance of data in the Cloud.
- Work-flow Layer: Monitoring the data flow and the work-flow of the data in the Cloud.
- Laws and Regulation layer: Compliance and alignment of data flow and evolution in the Cloud.
- Policies Layer: Compliance and alignment to internal policies.

Provenance can be collected from several data-centric logging mechanisms in the Cloud. One of such sources is Flogger [36], a distributed file-centric VM/PM logger which monitors file operations and transfers within the Cloud. Flogger addresses the TrustCloud System layer by capturing and logging who, what (file operation), at where (both PM and VM), and at what time a file is accessed, duplicated or transferred. Basic provenance data is collected from Flogger logs. A screenshot of the logs from Flogger is shown in Fig. 3; the file-centric logs from Flogger generate signatures and footprints, enabling us to track the provenance of files in and across both VMs and PMs in the cloud. With the logged atomic file actions, we can identify simple events such as an application doing an auto-backup. We are also able to distil heuristics for prevention of malicious events in the Cloud. We are currently working on categorising, studying and the types of atomic transactions and their combinations.



	filename text	full_path text	action text	date_time timestamp with time zone	file_username text	process_username text	vm_interface text	pm_ip4 text	pm_mac text	pid integer	p_uid integer	p_gid integer	timeval_sec bigint	timeval_usec bigint
1	aounts	/3350/aounts	Read	2011-03-23 14:00:26+08	tomcat	tomcat a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3350	500	500	1300860025	543880
2	install.txt	/home/tomcat/kernel trust/install.txt	Read	2011-03-23 14:00:34+08	tomcat	tomcat .:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3351	500	500	1300860033	219409
3	aounts	/3352/aounts	Read	2011-03-23 14:00:41+08	tomcat	tomcat .1:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3352	500	500	1300860040	457650
4	document.txt	/home/tomcat/kernel trust/test/document.txt	Create	2011-03-23 14:00:53+08	tomcat	tomcat :21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3353	500	500	1300860053	136480
5	aounts	/3356/aounts	Read	2011-03-23 14:01:01+08	tomcat	tomcat 59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3356	500	500	1300860060	300614
6	listing.txt	/home/tomcat/kernel trust/listing.txt	Read	2011-03-23 14:01:14+08	tomcat	tomcat :59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3367	500	500	1300860073	505171
7	document.txt	/home/tomcat/kernel trust/test/document.txt	Read	2011-03-23 14:01:24+08	tomcat	tomcr :9:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3368	500	500	1300860083	278415
8	document.txt	/home/tomcat/kernel trust/test/document.txt	Write	2011-03-23 14:01:28+08	tomcat	toa :29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3368	500	500	1300860087	485114
9	aounts	/3369/aounts	Read	2011-03-23 14:01:31+08	tomcat	tr	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3369	500	500	1300860090	574643
10	document2.txt	/home/tomcat/kernel trust/test/document2.txt	Create	2011-03-23 14:01:43+08	tomcat	tomcat /	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3370	500	500	1300860102	566805
11	aounts	/3371/aounts	Read	2011-03-23 14:01:49+08	tomcat	tomcat a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3371	500	500	1300860108	933135
12	aounts	/3374/aounts	Read	2011-03-23 14:02:01+08	tomcat	tomcat .:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3374	500	500	1300860120	786127
13	test.txt	/home/tomcat/kernel trust/src/test.txt	Read	2011-03-23 14:02:09+08	tomcat	tomcat .1:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3375	500	500	1300860128	272957
14	aounts	/3376/aounts	Read	2011-03-23 14:02:11+08	tomcat	tomcat :21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3376	500	500	1300860130	700398
15	install.txt	/home/tomcat/kernel trust/install.txt	Read	2011-03-23 14:02:22+08	tomcat	tomcat :9:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3377	500	500	1300860142	50178
16	aounts	/3379/aounts	Read	2011-03-23 14:02:30+08	tomcat	tomcat :59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3379	500	500	1300860149	806354
17	document.txt	/home/tomcat/kernel trust/test/document.txt	Read	2011-03-23 14:02:35+08	tomcat	tomcr :9:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3380	500	500	1300860155	123242
18	document.txt	/home/tomcat/kernel trust/test/document.txt	Write	2011-03-23 14:02:41+08	tomcat	toa :29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3380	500	500	1300860161	11626
19	document2.txt	/home/tomcat/kernel trust/test/document2.txt	Read	2011-03-23 14:02:45+08	tomcat	tr :29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3381	500	500	1300860164	678920
20	document2.txt	/home/tomcat/kernel trust/test/document2.txt	Write	2011-03-23 14:02:48+08	tomcat	0c:29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3381	500	500	1300860167	232444
21	aounts	/3382/aounts	Read	2011-03-23 14:02:48+08	tomcat	:0c:29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3382	500	500	1300860168	70155
22	ipaddress.c	/home/tomcat/kernel trust/test/ipaddress.c	Read	2011-03-23 14:03:02+08	tomcat	00:0c:29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3383	500	500	1300860181	358884
23	aounts	/3384/aounts	Read	2011-03-23 14:03:05+08	tomcat	00:0c:29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3384	500	500	1300860184	984029
24	split.c	/home/tomcat/kernel trust/test/split.c	Read	2011-03-23 14:03:07+08	tomcat	00:0c:29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3385	500	500	1300860186	965535
25	aounts	/3387/aounts	Read	2011-03-23 14:03:12+08	tomca	00:0c:29:59:21:a7	eth0	192.168.198.1	ao:7f:4d:b2:6d:c5	3387	500	500	1300860191	748271

Fig. 3. [36]: Provenance from Signatures in Flogger Logs.

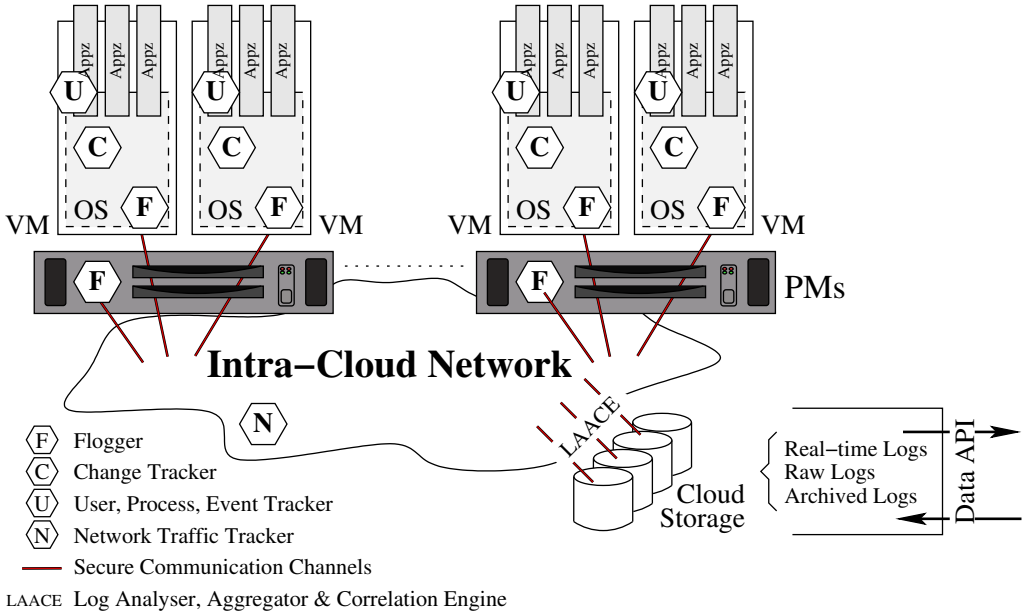


Fig. 4. DataPROVE Framework – Conceptual Overview.

The logs collected by Flogger form the basis for establishing a data-centric view of an entire Cloud environment. Figure 4 provides a schematic overview of our envisioned data provenance framework DataPROVE. This framework addresses the first four provenance granularity layers as defined in Section III; Inter-Cloud, Cloud-to-Internet and Internet-to-Cloud scenarios are not directly covered, but mitigated through the introduction of a DataAPI that permits signed/verifiable upload of data provenance information. Besides Flogger, the DataPROVE framework also contains the following tools:

- Change Tracker: Monitoring changes at the file system level. For instance, this tracker captures which blocks of a file have been modified or which records in a database table were changed.
- User, Process and Event Tracker: Monitoring user activities, processes and events. This complements Flogger's and Change Tracker's more file-container-centric logs and forms the basis for capturing provenance across the various layers (file system, operating system, processes, applications, user interactions) within a single machine.
- Network Traffic Tracker: Monitoring the movement of packets within the network. This complements Flogger's file-centric logs adding information about which files were sent from or received at a particular VM or PM and which path those files travelled. For instance, it will allow to capture the various network routes of all the packets belonging to the same file during a file transfer.

We are currently implementing the various trackers using ArcSight tools [37].

Considering the need for adequate protection of provenance data, all data collection mechanisms only communicate through secure data communication channels; DataPROVE's communication protocol is similar to PASS P3 and ensures data consistency, atomicity, causal ordering and persistence. During data collection, storage and archival processes, provenance data is undergoing various transformations (including correlation of provenance information collected from the various logger mechanisms, removal of redundant information,

data de-duplication, identification of common patterns and events, classification of data by expected usage (e.g., real-time versus forensic) etc.) designed to optimise storage as well as querying properties. This is facilitated by the LAACE module depicted in Figure 4. According to the classification/separation of collected provenance data, it is placed in the respective cloud storage-based real-time logs, raw logs, and/or archived logs; each of which is optimised to balance the trade-off between effective storage and efficient access. Indexing and querying facilities are provided through a well-defined Data API. This API also provides means to upload signed provenance information for data that is uploaded into the cloud together with adequate verification mechanisms.

VI. Concluding Remarks

In this paper, we emphasised on the necessity for better transparency and accountability of data managed in a cloud. We argued that adding data provenance mechanisms to cloud computing and storage offerings is a key requirement for a further increase and sustained adoption of cloud services. After identifying key challenges and requirements for collecting provenance in a cloud, we discussed our proposed approach, DataPROVE, that aims at enabling a provenance supplemented cloud for better integrity and safety of customers' data.

Moving forward, provenance collection must not only be restricted to a single cloud service provider's solutions. Instead, Inter-Cloud, Cloud-to-Internet and Internet-to-Cloud data movement and management scenarios also need to be investigated further.

Acknowledgment

The authors would like to thank the entire TrustCloud project team, for their insights on this research, and valuable comments on this paper.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[2] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance techniques," Computer Science Dept, Indiana University, Tech. Rep. IUB-CS-TR618, 2005.

[3] D. G. Clarke and D. M. Clark, "Lineage," *Elements of Spatial Data Quality*, 1995.

[4] J. L. Romeu, "Data quality and pedigree," *Material Ease*, 1999.

[5] H. V. Jagadish and F. Olken, "Database management for life sciences research," *SIGMOD Rec.*, vol. 33, pp. 15–20, 2004.

[6] Genepattern. Broad Institute. [Online].http://www.broadinstitute.org/cancer/software/genepattern/

[7] C. Pancerella, J. Hewson, W. Koegler, D. Leahy, M. Lee, L. Rahn, C. Yang, J. D. Myers, B. Didier, R. McCoy, K. Schuchardt, E. Stephan, T. Windus, K. Amin, S. Bittner, C. Lansing, M. Minkoff, S. Nijsure, G. von Laszewski, R. Pinzon, B. Ruscic, A. Wagner, B. Wang, W. Pitz, Y.-L. Ho, D. Montoya, L. Xu, T. C. Allison, W. H. Green, Jr., and M. Frenklach, "Metadata in the collaboratory for multi-scale chemical science," in *Proc of the Int'l Conf on Dublin Core and Metadata Applications*. Dublin Core Metadata Initiative, 2003, pp. 13:1–13:9.

[8] J. Frew and R. Bose, "Earth system science workbench: A data management infrastructure for earth science products," in *Proc of the 13th Int'l Confe on Scientific and Statistical Database Management*. IEEE Computer Society, 2001, pp. 180–189.

[9] Provenance aware service oriented architecture. [Online].http://twiki.pasoa.ecs.soton.ac.uk/bin/view/PASOA/WebHome

[10] I. Foster, "The virtual data grid: a new model and architecture for data-intensive collaboration," in *Proc of the 15th Int'l Conf on Scientific and Statistical Database Management*. IEEE Computer Society, 2003.

[11] The Kepler project. [Online].https://kepler-project.org/

[12] VisTrails. [Online].http://www.vistrails.org/index.php/Main_Page

[13] R. Bose and J. Frew, "Composing lineage metadata with xml for custom satellite-derived data products," in *SSDBM*, 2004, pp. 275–284.

[14] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware storage systems," in *Proc of the USENIX Technical Conf*. USENIX Association, 2006, pp. 43–56.

[15] First provenance challenge. [Online].http://twiki.ipaw.info/bin/view/Challenge/FirstProvenanceChallenge

[16] 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 *Proc of the USENIX Technical Conf*. USENIX Association, 2009, pp. 129–142.

[17] K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer, "Provenance for the cloud," in *Proc of the 8th USENIX Conf on File and Storage Technologies*. USENIX Association, 2010, pp. 197–210.

[18] R. Bose and J. Frew, "Lineage retrieval for scientific data processing: a survey," *ACM Comput. Surv.*, vol. 37, pp. 1–28, 2005.

[19] J. Vázquez-salceda, S. Álvarez, T. Kifor, L. Z. Varga, S. Miles, L. Moreau, and S. Willmott, "EU provenance project: an open provenance architecture for distributed applications," in *Agent Technology and e-Health*. Birkhäuser Verlag, 2008, pp. 55–64.

[20] I. M. Abbadi and J. Lyle, "Challenges for provenance in cloud computing," in *Proc of the 3rd USENIX Workshop on the Theory and Practice of Provenance*. USENIX, 2011.

[21] K.-K. Muniswamy-Reddy and M. Seltzer, "Provenance as first class cloud data," *SIGOPS Oper. Syst. Rev.*, vol. 43, pp. 11–16, 2010.

[22] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, 2010.

[23] (2010) Personal data in the cloud: A global survey of consumer attitudes. Fujitsu Research Institute. [Online].http://www.fujitsu.com/downloads/SOL/fai/reports/fujitsu_personal-data-in-the-cloud.pdf

[24] M. Mowbray, "The fog over the grimpen mire: Cloud computing and the law," *Scripted Journal of Law, Technology and Society*, vol. 6, 2009.

[25] D. Bhagwat, L. Chiticariu, W.-C. Tan, and G. Vijayvargiya, "An annotation management system for relational databases," *The VLDB Journal*, vol. 14, pp. 373–396, 2005.

[26] P. Buneman, S. Khanna, and T. Wang-Chiew, "Why and where: A characterization of data provenance," in *Database Theory*, ser. LNCS. Springer, 2001, vol. 1973, pp. 316–330.

[27] J. Widom, "Trio: A system for integrated management of data, accuracy, and lineage," in *Proc of the CIDR Conf*, 2005.

[28] C. Sar and P. Cao, "Lineage file system," Department of Computer Science, Stanford University, Tech. Rep., 2005. [Online].http://crypto.stanford.edu/ cao/lineage

[29] M. Sakka, B. Defude, and J. Tellez, "Document provenance in the cloud: Constraints and challenges," in *Networked Services and Applications - Engineering, Control and Management*, ser. LNCS. Springer, 2010, vol. 6164, pp. 107–117.

[30] 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.

[31] P. Macko, M. Chiarini, and M. Seltzer, "Collecting provenance via the Xen hypervisor," in *3rd USENIX Workshop on the Theory and Practice of Provenance*, 2011.

[32] Amazon web services. [Online].http://www.aws.amazon.com

[33] Amazon.com, "Amazon SimpleDB," http://aws.amazon.com/simpledb, 2009.

[34] R. K. L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "TrustCloud: A framework for accountability and trust in cloud computing," in *2nd IEEE Cloud Forum for Practitioners*. IEEE Press, 2011.

[35] J. Shende. (2010) Live forensics and the cloud - part 1. [Online].http://cloudcomputing.sys-con.com/node/1547944

[36] R. K. L. Ko, P. Jagadpramana, and B. S. Lee, "Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments," HP Labs Singapore, Tech. Rep. HPL-2011-119, 2011.

[37] ArcSight.com, "The ArcSight ETRM platform," http://www.arcsight.com/products/, 2011.