J Supercomput (2017) 73:1509–1531
DOI 10.1007/s11227-016-1838-0

Constructing data supply chain based on layered PROV

Peng Li[1] · Tin-Yu Wu[2] · Xin-Ming Li[3] · Hong Luo[1] · Mohammad S. Obaidat[4]

Published online: 12 September 2016

Abstract The inability to effectively construct data supply chain in distributed environments is becoming one of the top concerns in big data area. Aiming at this problem, a novel method of constructing data supply chain based on layered PROV is proposed. First, to abstractly describe the data transfer processes from creation to distribution, a data provenance specification presented by W3C is used to standardize the information records of data activities within and across data platforms. Then, a distributed PROV data generation algorithm for multi-platform is designed. Further, we propose a tiered storage management of provenance based on summarization technology, which reduces the provenance records by compressing mid versions so as to realize multi-

✉ Tin-Yu Wu
tyw@niu.edu.tw

Peng Li
13623351437@163.com

Xin-Ming Li
139117229321@163.com

Hong Luo
luoh@bupt.edu.cn

Mohammad S. Obaidat
msobaidat@gmail.com; m.s.obaidat@ieee.org

1 Beijing Key Lab of Intelligent Telecommunication Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China

2 Department of Computer Science and Information Engineering, National Ilan University, Yilan, Taiwan, ROC

3 Science and Technology on Beijing Complex Electronic System Simulation Laboratory, Academy of Equipment, Beijing 101416, China

4 Department of Computer and Information Science, Fordham University, New York 10458, USA

level management of PROV. In specific, we propose a hierarchical visual technique based on a layered query mechanism, which allows users to visualize data supply chain from general to detail. The experimental results show that the proposed approach can effectively improve the construction performance for data supply chain.

Keywords Data supply chain · Data platform · Provenance · PROV · Distributed environment

1 Introduction

In the area of big data, trading platforms for big data transform data into a saleable product which enables data exchange and data integration across different platforms and drives the prosperity of big data industry. Data trading platforms can accelerate data movement and usage among different enterprises. The supply chain begins when data is created, imported, or combined with other data; the data then moves, flows, and transforms through the supply chain [1]. Therefore, data supply chain can help break down the data silos and enables data to flow freely. As a result, organizations have the opportunity to ingest new sources of data; decision makers can gain the external information before making a decision. Besides, data supply chain also demonstrates the intact life cycle of data and identifies the publisher and subscriber accurately. Through such a chain, data may move swiftly from its source to places where it is needed. Although data supply chain can benefits all parts, existing tools can only demonstrate individual actions of independent platform, but cannot provide end-to-end vision of the entire chain across different platforms. This makes it impossible to sort out the relationships between seekers and providers effectively and hence impacts the integrality and efficiency on constructing data supply chain. Therefore, how to construct an end-to-end data supply chain in the distributed environment becomes a big challenge.

Provenance is the metadata that represents the history or lineage of a data object [2]. It is widely used in various areas, such as data rebuilding [3], debugging [4], and cloud security [5]. Provenance, which exactly describes the origin or lineage of a data node, shows the details of the creation of a data node and reveals the upstream and downstream between different data nodes. Therefore, provenance can play an important role in constructing the data supply chain. We cannot, however, directly apply existing provenance techniques to construct data supply chain, because data-at-rest and data-in-transit cannot be tracked together as a flow crossing system boundaries. Evidently, there needs to be an end-to-end tracking methodology for data provenance. Developing an end-to-end data supply chain system across platforms entails substantial issues. First, there is no uniform standard for data provenance in distributed environment. How to manage them effectively remains challenging. Second, as time goes by, the newly generated data will increase many folds which makes the data flow difficult to understand. There are seldom reported approaches that can achieve high effective querying and the structure of data supply chain still overwhelmingly complex.

In this paper, we propose a set of algorithms for constructing data supply chain based on layered PROV, and design the corresponding data supply chain construct-

ing system which can provide a zoomable visualization of data supply chain, called ACDSC-LP. Here, PROV is one widely used representation standard of provenance. The proposed ACDSC-LP system, by classifying provenance records into multiple levels, queries and consolidates the desired records according to users' requests in distributed environment, thus enables to replay data supply chain quickly. The main contributions of this paper are threefold.

(1) We design a construction system of data supply chain based on layered PROV to track data supply chain across different platforms. The proposed framework is able to robustly deal with the problem of managing disparate provenance records in distributed environments which cannot be dealt with by traditional PROV data model. (2) We propose a tiered storage management of provenance records based on summarization technology to enhance the effectiveness of data management by leveraging multi-level provenance records. (3) We propose a hierarchical visual technique based on a layered query mechanism, which allows users to visualize data supply chain from general to detail.

We conduct simulation experiments and the experimental results show that the proposed approach can effectively constructs an end-to-end data supply chain. Furthermore, the query performance outperforms the existing algorithms by at least 20 %.

The rest of the paper is organized as follows. We formally define the basic model of data supply chain in Sect. 2. We elaborate the design and implementation of our system framework in Sect. 3. In Sect. 4, We present the details of an algorithm for constructing data supply chain based on distributed PROV and a tiered storage management of provenance based on summarization technology, followed by the performance evaluation in Sect. 5. We summarize background and related work in Sect. 6. In Sect. 7, we conclude the paper.

2 Related work

In recent years, the usage of provenance [6] in different fields of computer science has been widely appreciated. Previous researches on provenance have mainly focused on workflow systems, operating systems, file systems and cloud computing. Xie et al. [3] used provenance, the origin or history of objects, to rebuild damaged or lost files. This method (PDRM) can exactly reconstruct the right file lost or damaged by back tracking its generation process in the past. During rebuilding, the method reveals the dependency between different nodes by replaying the generation process to display data supply chain. In our ACDSC-LP method, by merging and sorting provenance records across different data platforms, client gets the data supply chain. While PDRM method uses provenance reconstruct the ancestor node. Muniswamy-Reddy et al. [7] analyzed the provenance collected from multiple workloads with a view towards efficient storage. Based on the analysis, they characterize the properties of provenance with respect to long term storage. Then they propose a hybrid scheme (HA-WCDE) that explores the locality and similarity characteristics among the entity nodes to look for frequently occurring strings, and then use integer codes to encode them. After that, the method eliminates the duplicate strings that achieved the best compression ratio. In our TSMP-ST method, the stored mid versions can be retrieved through querying PROV database. We compressed these nodes and import them into Data Archive data-

base. To allow provenance information to be exchanged between systems, Moreau et al. [8] proposed a shared provenance model, the Open Provenance Model (OPM). In OPM, provenance graphs consist of three types of nodes. Artifacts represent an immutable piece of state, which may have a physical embodiment in a physical object, or a digital representation in a computer system. Processes represent actions performed on or caused by artifacts, and resulting in new artifacts. Agents represent contextual entities acting as a catalyst of a process, enabling, facilitating, controlling, or affecting its execution. PROV [9] is a conceptual data model that forms a basis for the W3C provenance family of specifications. PROV distinguishes core structures, forming the essence of provenance information, from extended structures catering for more specific uses of provenance. Ryan et al. [5] presented Progger (Provenance Logger), a kernel-space logger which potentially empowered all cloud stakeholders to trace their data. However, the technology does not collect the provenance of objects on multiple granularities, users would be unable to detect intrusions or perform forensic analysis on distributed systems. Jones et al. [10] detailed a provenance enabled file system that automatically collected information flow provenance at the filesystem level with the goal of aiding scientific users in better organizing their data. However, the approach collected provenance by intercepting system commands at the kernel level, which added additional overhead. Mattoso et al. [11] surveyed the early and current efforts in dynamic workflows and user steering and proposes a taxonomy to identify the main concepts related to addressing issues in dynamic steering of high performance computing (HPC) in scientific workflows. However, those approaches did not provide an interface for users to retrieve the recorded provenance information. Without such an interface, even the most detailed provenance will be just a white elephant. Korolev et al. [12] proposed a tool that aided researchers to improve reproducibility of their experiments through automated keeping of provenance records. Imran et al. [13] proposed an intuitive layer based architecture of data provenance and visualization. In addition, they show a complete workflow of tracking provenance information of big data.

SPADE [14,15] is a developing file system support to transparently generate and certify distributed data provenance, and a corresponding tool to validate it. However, there is no uniform standard for data provenance record until now and SPADE cannot establish such relationships between seemingly disparate events in the provenance records. Suen et al. [16] introduced S2Logger, a data event logging mechanism which captures, analyzes and visualizes data events in the cloud from the data point of view. Jacobson et al. [17] have designed and implemented a sharing system that does not require infrastructure yet supports robust, distributed, secure sharing by opportunistically using any and all connectivity, local or global, permanent or transient, to communicate. Information sharing is a key element in any Supply Chain Management (SCM) system and is critical for improving supply chain performance and enhancing the competitive advantage of an organization. However, many organizations are reluctant to share information with their supply chain partners because of lack of trust, the fear of information leakage and security attacks from malicious individuals or groups. Zhang et al. [18] examined the possible security threats/attacks in a SCM system. Then the key technologies and techniques for securing the information shared in SCM are identified with the goal of improving organizations' capabilities in sharing secure information.

While the authors have identified several related pitfalls in their proposed approach, there are still gaps with respect to providing complete data supply chain within and across data platforms. To the best our knowledge, our work is the first successfully constructed data supply in distributed environments using provenance technology.

3 Model and terms of data supply chain

Data supply chain depicts the actions performed on data and the entities being responsible for those actions throughout its life cycle. Data's life cycle can consist of creation, usage, transform and distribution of the data. The conceptions of the model of data supply chain are introduced in the following:

1. Entity node
Data supply chain enables data to flow freely. *Entity nodes* in data supply chain is a set of data or file physically stored in a data platform. Generally, an entity node could be recorded representing multiple sources, or just representing a single data source. One entity node might be related to other entity nodes. For instance, if a document D is derived from a data table T, then both D and T are entity nodes while T is the direct ancestor node of D and D is the successor node of T.

2. Activity
Activities describe how entity nodes come into existence and how their attributes change. For instance, the process of translating a data table into another language, which creates a new data table, is called an activity.

3. Agent
An *agent* is a person or an organization who has responsibility for an activity. The relationship between agent and its corresponding activity is called *association*. Several agents can be associated with the same activity. An entity node involved in the activity is attributed to the related *agent*. For instance, an activity tabulates the data into a table. The person who has responsibility for the tabulation is the agent that is associated to this activity. And the table is the entity node that attributed it to this agent.

4. Generation and usage
Generation is the production of a new *entity node* by an *activity*. For instance, the activity of creating a data table generates a new entity node called data table. Usage is the utilization of an entity node by an activity. For instance, correcting the spelling mistakes uses the former edition. Provenance can exactly describes the origin or lineage of an entity node, shows the details of the creation of an entity node and reveals the dependency between different entity nodes. Therefore, Provenance can be used as an important clue to construct data supply chain. Currently, the prevalent standards of data provenance models are OPM [19] and PROV [9]. The PROV specification presented by W3C gives a standard description of data provenance. It has strong analyticity and semantic features and can be recorded in the forms like XML, JSON, OWL2 and etc, which offers great convenience in operations. Meanwhile, the PROV specification also presents the standardized and readable description format, called PROV-N. What's more, PROV defines inference rule to consolidate provenance

information, which makes it possible to construct data supply chain and achieve data provenance among distributed platforms. As a conclusion, we choose PROV-N as the provenance standard in our data supply chain model.

Let us see a typical application usecase shown in Fig. 1 suppose that there are three information platforms for companies: A, B and C. Platform-A extracts a part of Data 1 to generate Data 2. Data 2 are modified and converted to Data 3 using weighting. Data 3 flows from Platform-A to Platform-B. Platform-B adds Data 3 and Data 4, and assigns the result to Data 5. Data 5 flows from platform B to C. After being disposed by C, Data 5 turned into Data 6 and then be revised for twice.

In this data supply chain, Data 1, Data 2, Data 3, Data 4, Data 5 and Data 6 are entity nodes. We assign each entity node a unique number, called data ID. Data ID consists of prefix and sequence number which is used to identify different entity node. Referring to PROV-N, in which prefix serves as the organizations that entities belong to, we define the above data (Data1 to Data 6) as six entity nodes:

Entity(A-1), Entity(A-2), Entity(A-3),
Entity(B-1), Entity(B-2) and Entity(C-1)

In this data supply chain, the processes of disposing data, executed by the data Platform-A, Platform-B and Platform-C are activities. For example, Process Sum is activity executed by Platform-B at time T_B. Referring to PROV-N, we define activity Process Sum as follows.

Activity (Platform B: Process Sum, T_B)

Agents record the information of agents and its relationships with entity node and activity. In this data supply chain, the activity Process Sum is executed by Platform-B. Therefore, organization B is an agent. Referring to PROV-N, we define an agent as follows.

Agent(Platform B, [prov:type='Organization', foaf:givenName="B"])

The relationship between activity and entity node is described by generation and usage. For instant, in this data supply chain, Platform-B adds Entity (A-3) and Entity (B-1), and assigns the result to the Entity (B-2), T_B is the occurrence time. Referring to PROV-N, we define them as follows:

Used (Platform B: Process Sum, A-3)
Used (Platform B: Process Sum, B-1)
wasGeneratedBy (B-2, Platform B: Process Sum, T_B)

From what was described above, in this data supply chain, we describe the entity node B-2 completely as follows.

Entity(B-2)

Activity (Platform B: Process Sum, T_B)
Agent (Platform B,[prov:type='Organization' foaf:givenName="B"])
Used (Platform B: Process Sum, A-3)
Used (Platform B: Process Sum, B-1)
wasGeneratedBy (B-2, Platform B: Process Sum, T_B)

4 System framework

Figure 2 shows the framework of construction system of data supply chain based on layered PROV. As depicted in Fig. 2, data exchange system, data processing system and data supply chain management system are deployed in each data platform, where data supply chain management system consists of three modules, namely, the PROV data generator, PROV data reorganizing module and data supply chain extraction module. Data platforms provide data intensive computation such as activity recognition and sentiment analysis for integration of various multi-structure data and its processing. Supercomputers use and generate large amounts of data and having its provenance is valuable. Through the distributed deployed data supply chain management system and related communications between upstream and downstream data platforms, the visualization of data supply chain is provided according to user's request.

1. *PROV data generator* It is responsible for getting attribute arguments which depicts the actions performed on data and the entities being responsible for those actions. Each PROV record, which contains identity information, activity, occurring time, agent and related upstream and downstream, is stored in the PROV database as an entity node. On this basis, we can construct dynamic data supply chain.

2. *PROV data reorganizing module* The amount of provenance information about data products are rapidly growing throughout its life cycle. Relying on the numerous and jumbled provenance records to build data supply chain, the data flow structure is overwhelmingly complex. PROV Data Reorganizing Module moves mid versions into the data archive database and stores the collected provenance in a hierarchical structure so as to manage provenance data with high efficiency.

3. *Data supply chain extraction module* (*DSCEM*) DSCEM is responsible for receiving user's query request, performing local processing and forwarding the query, etc. Each DSCEM communicates with its related DSCEMs in upstream or downstream data platforms, so as to obtain the related PROV records on the required chain for visualization.

There are two stages in the procedure of constructing and visualizing the data supply chain, namely PROV data generation stage and data supply chain query stage. In the first stage, for the data has been published by other data platforms, we download and store them into a DataSource database. We assign each raw data a unique num-

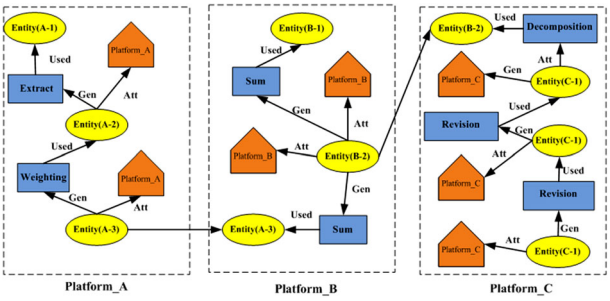

Fig. 1 Example of data flow among data platforms

ber (called data ID), then PROV records describing source of data and the mapping relationship between data name and its data ID are written in the tables of PROV database. For the data is derived from a raw data which has been used by users, we also assign each derived data a unique number, for the PROV records describing the provenance of derived data, such as agent, activity and ancestor, we write them in other tables of PROV database. In the second stage, first, user submits a data chain query including the data ID. After DSCEM receives the query, it will visit local PROV database to get local provenance records. Then DSCEM generates and forwards some new queries to its related upstream and downstream DSCEMs according to the provenance records. Finally, DSCEM will merge and sorts all results from different data platforms, produces the final result, and shows user the desired data supply chain.

We set a data coherency unit for maintaining data coherency across a number of data platform. The data coherency unit monitors data transition states when the processing status of data being shared by two or more data platforms. The data coherency unit ensures a status change in shared data in one data platform, which is broadcast to other data platform having copies of the data without having each data platform independently monitor to detect data state changes. However, implement of data coherency unit is out of the scope of this paper.

5 Construction and visualization of data supply chain

As illustrated in the system framework, each data platform collects and stores provenance of data, respectively, when data flows across different data platforms. To solve the problem that there is no uniform standard for data provenance record and achieve cross-platform provenance query, we propose a PROV data generation algorithm for multi-platform in subsection IV.A, which uses PROV specification to translate provenance traces into the PROV records. Further, in subsection IV.B, we present an algorithm of constructing data supply chain based on distributed PROV. In this algorithm, by leveraging the uniform description of data provenance, a distributed query mechanism is used to build data supply chain. In addition, to query data supply chain with high efficiency, we propose a tiered storage management of provenance based on summarization technology in subsection IV.C, which can support more elaborate storage and provides more efficient processing on PROV records.

5.1 PROV data generation algorithm for multi-platform

To achieve cross-platform provenance trace and improve the performance of querying, we design six tables in PROV database: DSInfo, PROVInfo, DirectAnc-Info, SuccessorInfo, UPInfo and DPInfo.

Table 1 is responsible for storing the mapping relationship between the name of an entity node and its data ID. As discussed above, we have assigned a unique ID to each raw data or derived data when it is imported in or processed on the data platform. So the structure of DSInfo with five fields is (ID, data ID, entity name, Is original, Is succeed), where ID is the primary key of the table, Data ID is the identifier of an

Table 1 DSInfo

ID	Data ID	Table name	Is original	Is succeed
1	A-1	CPI	0	1
2	A-2	Product cost	1	1
3	A-3	Profits	1	0

Table 2 UPINFO

ID	Data ID	Source	URL
1	A-1	Nation	http://www.stats.gov.cn/

Table 3 PROVinfo

ID	Entity	Activity	Agent	Used	Time
1	A-2	Extract	Agent A	A-1	2015-5-15
2	A-3	Weighting	Agent A	A-2	2015-5-15

Table 4 DirectAnc-Info

ID	Data ID	Direct ancestor
1	A-2	A-1
2	A-3	A-2

entity node, Entity Name is the name of entity node, IS Original denotes whether the node has ancestor, IS Succeed denotes whether the node has successor. For example, if Is succeed = 1, it will forwards the query to the related downstream PROV records when receiving user's query request. If Is succeed = 0, it will not forwards the query because there is no successor.

Table 2 is responsible for storing all upstream platforms using the entity node. The structure of UPInfo with four fields is (ID, data ID, source, URL), where ID is the primary key, Data ID is the identifier of an entity node, Source is the identifier of upstream platforms, URL is the Uniform Resource Locator of upstream platforms.

Table 3 is responsible for storing the basic provenance information of entity nodes. Based on the PROV specification, the structure of PROVInfo with six fields is (ID, entity, activity, agent, used, time), where ID is the primary key, Entity is the data ID of entity nodes, Agent is the person or organization that has responsibility for an activity on the data, activity is the action, used is the data ID of its direct ancestor, and Time remarks when the activity occurs.

Table 4 is responsible for storing all direct ancestors of each entity node. The structure of DirectAnc-Info with three fields is (ID, data ID, direct ancestor), where ID is the primary key, data ID is the identifier of an entity node, direct ancestor is the identifier of one direct ancestor.

Table 5 is responsible for storing all the successors of an entity node. The structure of SuccessorInfo with three fields is (ID, data ID, successor), where ID is the primary key, data ID is the identifier of an entity node, successor is the identifier of its successor.

Table 6 is responsible for storing all downstream platforms using the entity node. The structure of DPInfo with four fields is (ID, Data ID, Next, URL), where ID is the primary key, Data ID is the identifier of an entity node, Next is the agent identi-

Table 5 Successorinfo

ID	Data ID	Successor
1	A-1	A-2
2	A-2	A-3

Table 6 DPInfo

ID	Data ID	Next	URL
1	A-3	B	http://www.datatang.com/

fier of downstream platforms, URL is the Uniform Resource Locator of downstream platforms.

For instant, the initial value of all tables in PROV database is empty. Platform-A gets Consumer Price Index (CPI) from National Statistic Bureau. Hence, we should store CPI into a DataSource database and write the first record in the table of DSInfo. At this time, only the first record with ID = 1 exists, the fields IS Original and IS Succeed of A-1 are 0 and 0. Besides, we write the first record in the table of UPInfo as shown in Table 2. Then, we extract a part of CPI to generate Product Cost. Now, we should store product cost into a DataSource database and add the second record into the table of DSInfo. Besides, the fields IS Original and IS Succeed of A-1 should change to 0 and 1, and the fields IS Original and IS Succeed of A-2 are 1 and 0. Next, We should write all the first records in the table of PROVInfo, DirectAnc-Info and SuccessorInfo as shown in Tables 3, 4 and 5 respectively. After weighting Product cost, we should store derived data Profits into a DataSource database and add the third record into the table of DSInfo. At the same time, the fields IS Original and IS Succeed of A-2 should change to 1, and the fields IS Original and IS Succeed of A-3 are 1 and 0. Later, We should add all the second records into the table of PROVInfo, DirectAnc-Info and SuccessorInfo, respectively. If the data flows from Platform-A to Platform-B later, we need to write a record in the table of DPInfo as shown in Table 6. In the PROV data generator on each data platform, there runs a SPADE system [14] which can collects the provenance traces depicting the actions performed on data flow and stores the information (e.g., identity information, activity, occurring time, agent and related upstream and downstream) in TraceInfo which is a text file. Then, the PROV data generation algorithm translates them into the PROV records and stores them in respective PROV database. Algorithm 1 gives the details of the PROV data generation algorithm.

The time complexity of the Algorithm 1 is $O(P * Q)$, where P is the whole number of node in XML file and Q is the amount of property of each node. The space overhead of the algorithm is $P * Q * K$, where K is average size of each property-value. Obviously, the runtime of the algorithm is directly affected by P and Q. The reason is that along with the scaling-up number of node in XML file, the amount of property displays a upward tendency, which increases the number of writes against the database and accordingly improves the runtime.

5.2 Constructing data supply chain based on distributed PROV

Visualization of data supply chain shows the data flow across different platforms in a graphical way, which significantly enhances the information transparency and

improves the efficiency of data supply chain management. To achieve visualization of data supply chain, we can get provenance records across different data platforms, then merge and sort the results and import them into a PROV Toolkit [9]. Based on the distributed PROV databases and records representing the links among entity nodes, any platforms under request can communicate with its ancestor/successor platforms to get the PROV records and construct the data supply chain. Further, data supply chain can be built through a command line interface linking together the provenance records from different data platforms. Let *queryID* denotes the unique query identifier, *DSC Results* denotes the data supply chain. The algorithm of constructing data supply chain based on distributed PROV is shown as Algorithm 2.

Algorithm 1 PROV data generation algorithm for multi-platform

Input: *TraceInfo*
Output: *PROV Info*, *DirectAnc Info*, *SuccessorInfo*, *DPInfo*, *UPInfo*

1: $XML \leftarrow$ writeXML(TraceInfo);
2: correlation attributes of node to each column of PROVInfo;
3: **for** each node $\in XML$ **do**
4: $Entity$ = node.InnerXml; $Activity$ = node.InnerXml;
5: $Agent$ = node.InnerXml; $Time$ = node.InnerXml;
6: $Used$= node.InnerXml;
7: **end for**
8: Write attribute-value to PROVInfo();
9: correlation attributes of node to each column of $DirectAnc - Info$;
10: correlation attributes of node to each column of $SuccessorInfo$;
11: **for** each node $\in XML$ **do**
12: $Direct\ \ Ancestor$ = node.InnerXml;
13: $Successor$ = node.InnerXml;
14: **end for**
15: Write attribute-value to DirectAnc-Info();
16: Write attribute-value to SuccessorInfo();
17: @Column(name="Next");
18: @Column(name="Source");
19: **for** each node $\in XML$ **do**
20: $Next$ = node.InnerXml;
21: $Source$ = node.InnerXml;
22: **end for**
23: Write attribute-value to DPInfo();Write attribute-value to UPInfo();
24: return *PROV Info*, $DirectAnc - Info$, $SuccessorInfo$, $DPInfo$, $UPInfo$;

The time complexity of Algorithm 2 is $O(M * N + N)$, where M is the average number of provenance records in database and N is the average number of nodes belonging to the same data supply. Obviously, the construction time of data supply chain is directly affected by M and N. To query the upstream and downstream of the node, there is a need to iterate over all of the records in PROV database. Along with the scaling-up number of M, query time displays a upward tendency, which increases the construction time of data supply chain; otherwise reduces the construction time. This indicates that maintaining a small provenance database can significantly benefit the construction performance. The whole construction time scales linearly as the time costs on querying database, while querying overhead shows improvement along with the increasing number of N. We will evaluate how they impact the whole construction performance in Sect. 5.

We load the final query results to the PROV Toolkit [19], the PROV Toolkit will automatically associate the query results and visualize them. Figure 3 shows the visualization result of the data supply chain.

Algorithm 2 Constructing data supply chain based on distributed PROV

Input: *queryID*, *PROVInfo*, *SuccessorInfo*, *DPInfo*, *UPInfo*, *DSInfo*, *DAInfo*
Output: *DSC Results*

```
obtaining queryID;
repeat
    using Data ID as the key, query the corresponding value from PROV Info;
    repeat
        using Data ID as the key, query the corresponding value;
        if IS Original = 1 then
            get provenance records representing ancestor;
        end if
    until IS Original = 0
    using Data ID as the key, query the corresponding URL;
    creat queryID;
until URL = null
likewise, query the downstream node;
return DSC Rsults;
```

5.3 Advanced scheme of constructing data supply chain based on layered PROV

In big data environment, there are a large number of provenance records representing the corresponding entity nodes, which incurs huge space and searching overhead even with distributed storage and searching scheme. By further study of the content of entity nodes, we found that there are huge number of entity nodes describing the mid version

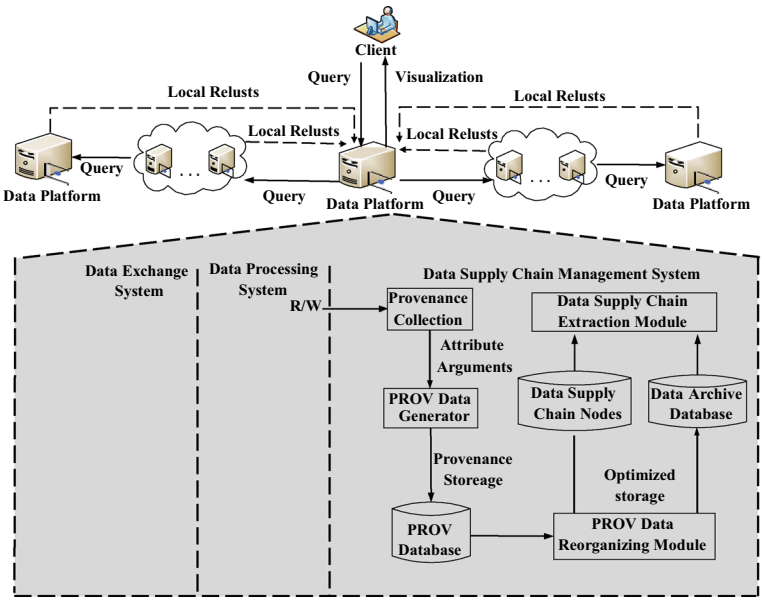

Fig. 2 Architectural overview of ACDSC-LP

or inner change of data, and those provenance information maybe not meaningful to the end user. Hence, a complex chain with so much detailed information may confuse the end user and difficult to traces. For example, to get the final result, researchers sometimes use software tools to revise the same experimental data set repeatedly during science experiments. This makes the experimental data sets having a lot of versions, where there is plenty of redundancy between final versions and mid versions, and only the final version of the data set will be used by other researchers. Hence, a zoomable data supply chain is more benefit to users, which can not only include the flow of final version but also reflect the subtle changing of data. To solve the problem flexibly, we propose an advanced scheme of constructing and visualizing data supply chain based on layered PROV, which may may extract the major contents from original entity nodes to create an overview of the chain and store the records of mid versions in data achieve database for further querying. In addition, we design a certain threshold F to control how many detailed records should be hidden, so as to realize multi-level management of PROV. In the next of this section, we first describe the method of layered PROV management based on summarization technology, and then present the algorithm of constructing a refined data supply chain through a layered query mechanism.

1. *Layered storage management of PROV* To compress the entity nodes with mid version, we propose a layered storage management of PROV based on summarization technology, which allows user to search the node with changing attributes while still not affecting the integrality of data supply chain. The basic idea of this algorithm is to compress PROV records with unimportant versions and thus significantly reduce the storage overhead of first level PROV records. To make it more convenient when querying detailed compressed PROV information, we build an index for storing the

relationship between the identifier of a PROV record and its storage location in the data archive database storing all the entity nodes with mid version. Let F denotes the upper threshold of the maximum number of mid version contained by a node. If a node contains more number of mid version than the threshold F, the provenance records representing current node will be compressed. Let D denotes the original provenance sets, $D\ target$ the denotes target provenance sets for compressing, $AI\ Database$ denotes data archive database. The procedure of compressing an original PROV database to a layered one based on summarization technology is shown as Algorithm 3.

The time complexity of Algorithm 3 is $O(M + n * (N^2 + N))$, where M is the overall number of records in provenance sets, N is the overall number of records in target provenance sets and n is the average number of node. The space overhead of the algorithm is $n + N * S$, where S is average size of each provenance record. Obviously, the compression perform is directly affected by F. The reason is that N is determined by F. Along with the scaling-up F, N displays a downward tendency, which reduces the compression number and decreases the runtime; otherwise improves the runtime. The provenance query perform is directly affected by n. This is because we impose smaller n leads to larger target provenance sets for compressing when N is fixed. Therefore, query perform shows improvement along with the decreasing n. We will evaluate how they impact the query and compression performance in Sect. 5.

Algorithm 3 Tiered storage management of provenance based on summarization technology

Input: D, F
Output: compressed D

```
1: //obtain the number of entity node ;
2: j <-- COUNT(DISTINCT Data ID) from D;
3: for i = 1 to j do
4:     N[i] <-- COUNT(Data ID);
5: end for
6: if N[i] > F then
7:     insert correlative Data ID INTO D target;
8: end if
9: //compressing the target provenance sets;
10: for each Data ID ∈ D target do
11:     sort the Data ID.allrecords based on time;
12:     Data ID.head.prev.next = Data ID.tail;
13:     Data ID.tail.prev = Data ID.head.prev;
14:     AI Database <-- Data ID.leftrecord;
15:     creat index table in AI Database;
16:     delete the Data ID.leftrecord;
17:     modifying Data ID.Activity;
18: end for
19: D <-- D target + D remainder
20: return D
```

2. *Visual process of multi-level data supply chain* Based on the layered storage of PROV records, to achieve multi-level visualization of data supply chain, we exploit a layered query mechanism to get PROV records from related data platforms, then merge and sort the results and import them into the PROV Toolkit. Since Algorithm 2 has achieved the goal of constructing the data supply chain according to users' requests in the distributed environments, we use Algorithm 2 as the core of the layered query mechanism. When user wants a general overview of the data supply chain, DSCEM runs Algorithm 2 to get the first level PROV records and load the results into the PROV Toolkit for visualization; when user wants to replay the data supply chain containing mid versions of special segment, DSCEM generates a new query with the Data ID of special entity node, then it communicates with the corresponding DSCEM and gets the PROV records in that local data archive database. Finally, DSCEM merges and sorts the results and gets the second level data supply chain only containing the mid versions of the special segment. Let *queryID* denotes first-query identifier, *DSC results* denotes the data supply chain, *DSC model* denotes visualized data supply chain, *Arch-queryID* denotes second-query identifier, *MDSC results* denotes data supply chain only containing mid versions, *MDSC model* denotes visualized data supply chain containing mid versions, *CDSC-DP* denotes the method of constructing data supply chain based on distributed PROV. The Visual algorithm of multilevel data supply chain is shown as Algorithm 4.

Algorithm 4 Visual algorithm of multi-level data supply chain

Input: *queryID*, *Arch-queryID*
Output: *DSC Results*, *DSC Model*, *MDSC Results*, *MDSC Model*

```
1: // send the first-query;
2: obtaining queryID;
3: Data ID←−analyzing(queryID);
4: DSC Rsults←−CDSC-DP(Data ID);
5: // visualization of data supply chain;
6: List provRecords←−DSC Rsults;
7: for i=0 to provRecords.size() do
8:     Prov provRecord = provRecords.get(i);
9: end for
10: return DSC Model;
11: Arch-queryID ←− analyzing(DSC Model);
12: using Data ID as the key, query the storage location
    from Index Table;
13: MDSC Rsults←−Query storage location from data
    archive database;
14: for j=0 to provRecords.size() do
15:     Prov provRecord = provRecords.get(j);
16: end for
17: return MDSC Model;
```

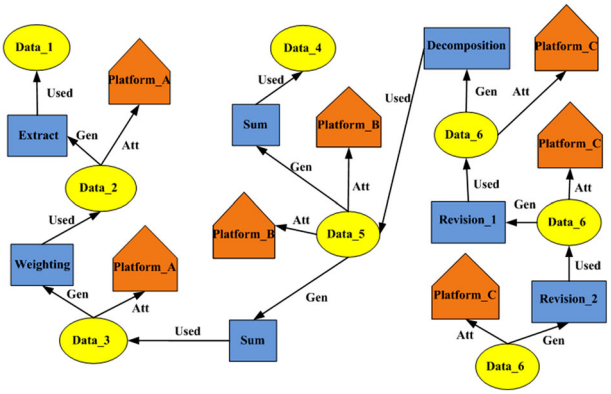

Fig. 3 Visualization of data supply chain

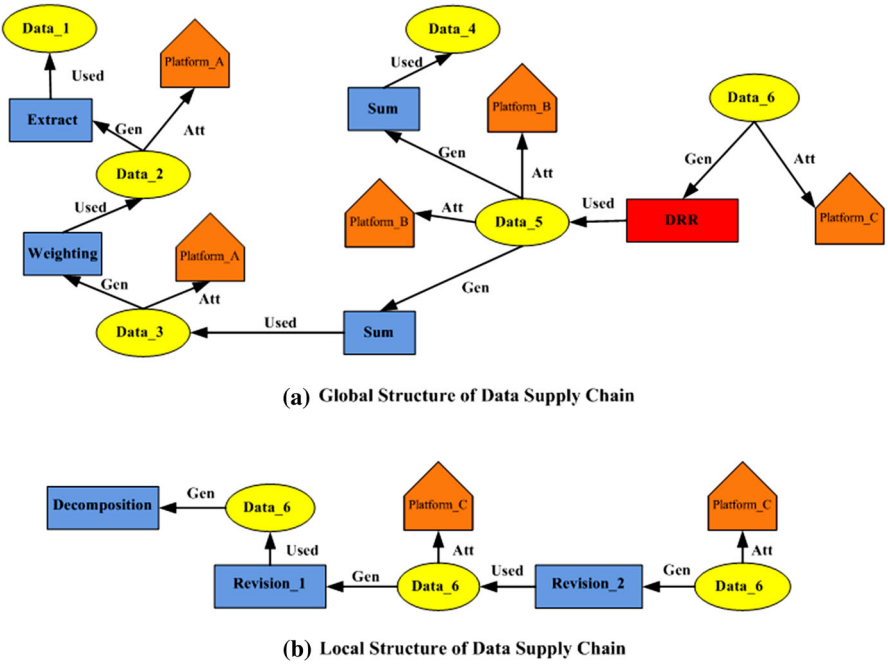

Fig. 4 Visual querying process of multi-level data supply chain

We use the same application usecase shown in Fig. 1 to verify the Algorithm 4. Figure 4 shows the layered data supply chain through Algorithm 4, where DRR denotes a representative from a set of the actions performed on mid version of Data 6. Compared with Fig. 3, the redundancy mid versions of Data 6 is not appeared in the overview of data supply chain, which makes the structure concise and clear. And we can see the detailed evolution of Data 6 in local chain.

In Algorithm 2, the construction time of data supply chain is directly affected by M and N. The construction time is composed of the time costing on provenance query and on construction execution. The construction time of the Algorithm 4 decreases a lot. The reason for the improvement on construction time is twofold. First, In Algorithm 4, we compress the provenance records in PROV database, which decreases M. Then the scaling-down queries leads to less query time. Second, the time costing on provenance query decreases linearly as the number of querying database reduces. The reason is that a layered query mechanism can make the node number of data supply chain less.

6 Experiments and analysis

6.1 Experimental setup

We run our experiments on several data platform simulators. The configurations of those simulator are Ubuntu15.04 operating system with Inter Core i5-3200M 2.5 GHz processors, 2 GB memory and 500 GB hard drive. We use the java programming language to implement the proposed algorithms. They are implemented in our own, custom library but the test data is not available. We collect provenance traces and translate them into the provenance records TraceDB. TraceDB depicts the actions performed on data when data flows across different data platforms. The size of provenance records in the experiments is up to 500 MB and we store them in a PROV database, data supply chain nodes database, and data archive database.

To verify the compression performance of the proposed algorithm, we compare our Tiered Storage Management of Provenance Based on Summarization Technology (TSMP-ST) with a Hybrid Approach that Combines Web Compression and the Dictionary Encoding (HA-WCDE) [7] from compression ratio. compression time and query time. Compression ratio is a value that represents the ratio of the volume from the size of the compressed file to uncompressed file size, the smaller the better. To verify the construction performance of the proposed algorithm, we compare our advanced algorithm of constructing data supply chain based on layered PROV (ACDSC-LP) with a provenance-based data reconstruction method (PDRM) [3] from query time and construction time.

6.2 Compression ratio and compression time

First, we compressed provenance records in a PROV database using our TSMP-ST algorithm and HA-WCDE algorithm, respectively. In our TSMP-ST algorithm, we compressed these nodes by moving mid versions into the data archive database. While HA-WCDE method explores the locality and similarity characteristics among the entity nodes to look for frequently occurring strings, and then use integer codes to encode them. After that, the method eliminate the duplicate strings to achieve the best compression ratio. ArtifactDB is responsible for storing entity node identifier. DictionaryDB is responsible for storing the mapping relationship between integer codes and frequently occurring strings.

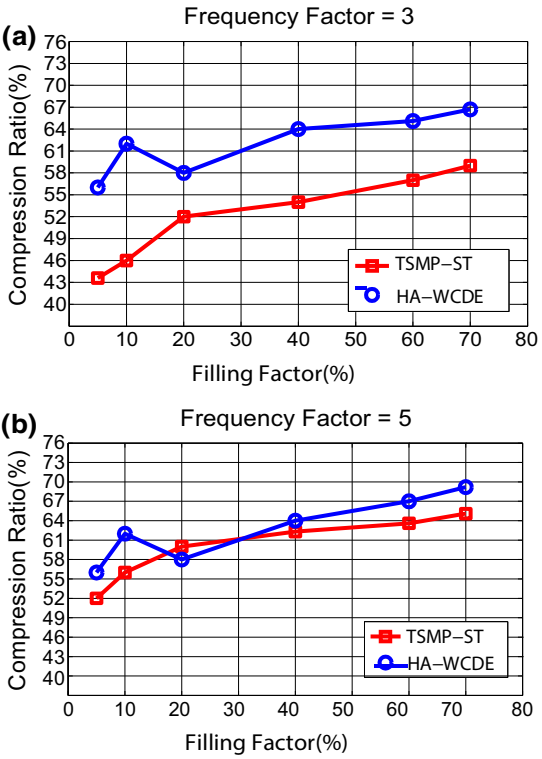

Fig. 5 Compression ratio for various filling factor

We set the frequency factor F to 3 and 5, where F is the upper threshold of the maximum number of mid version contained by a node. Due to most of nodes in the data database contained 3 or more numbers of mid version, so the value of F being set between 3 and 5. If a node contains more number of mid version than the threshold F, the provenance records representing current node will be compressed. For example, when the threshold F is 3 and a node contains 6 mid versions, the provenance records representing current node is 6. After compressing the provenance records, the numbers will be 2. So the results are dependent on the parameter F. We set different filling factor n (5, 10, 20, 40, 60, 70 %), where n is the percent that final versions taking up a portion of overall entity nodes. Figure 5 shows the compression ratio for various filling factor using TSMP-ST and the HA-WCDE methods respectively. This is because n is the percent that final versions (having no mid versions for compression) taking up the whole nodes. Along with the scaling-up n, the number of mid versions displays a downward tendency, which decrease provenance records for compressing so as to rise compression ratio gradually. In fact, the higher compression ratio means that smaller provenance sets for compressing. Therefore, increasing number of n will help improve the compression ratio".

From the results reported in Fig. 5, the compression ratios of TSMP-ST and HA-WCDE increase with increasing of filling factor. The compression ratio of TSMP-ST with $F = 3$ decreases by 6–12 % compared to compression ratio of HA-WCDE. The

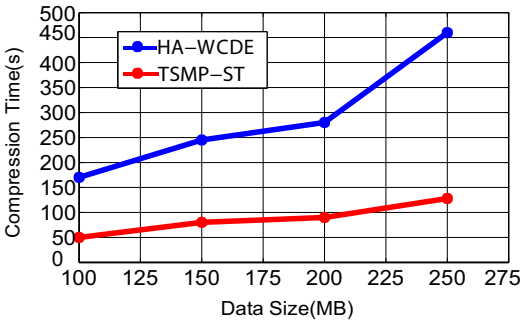

Fig. 6 Compression time for various data size

reason is that HA-WCDE scanning the entire database or text files to find the frequently occurring strings, and then replacing them with integer codes so as to eliminate any repeated separate strings. However, for the case using provenance to construct data supply chain, we only collect basic provenance which include node identity, ancestor, time and do not provide more detailed information of input parameters needed during the process execution and environment variables, etc. Compared with the previous results, HA-WCDE has been an obvious performance drop of eliminating the duplicate strings.

As shown in Fig. 5, the compression ratio of HA-WCDE displays a downward tendency. This is because HA-WCDE encodes the whole provenance sets using integer codes and the compression ratio is not affected by F. But we can decrease compression ratio by setting a reasonable F.

Figure 6 shows the compression time with data size varied from 100MB to 250 MB and the optimal setting of $F = 3$ and $n = 20$ % using TSMP-ST and HA-WCDE methods respectively.

From the results reported in Fig. 6, the time overhead of TSMP-ST reduced by 62–67 % compared to the time overhead of HA-WCDE. The reason for the improvement is two-fold. First, HA-WCDE uses integer codes to encode frequently occurring strings among the entity node and the mapping relationship in DictionaryDB, which increases compression time. Subsequently, the identifier of each node is encoded using coded value of ancestor, which can incur big time overhead during the ancestor queries. Second, for the TSMP-ST case, provenance records of PROV database in different data platforms are compressed concurrently in distributed environment. Hence the compression time in the TSMP-ST is much smaller than in the HA-WCDE case.

6.3 Query time

We ran two types of queries on the compressed provenance sets.

1. Query 1 looking up the direct ancestor of a specific entity node.
2. Query 2 looking up all the successors of a specific entity node.

From the results reported in Fig. 7, the query time of ACDSC-LP is less than that of HA-WCDE. Compared to HA-WCDE, ACDSC-LP saves at least 10 % time overhead. The reason for the improvement is two-fold. First, for the ACDSC-LP

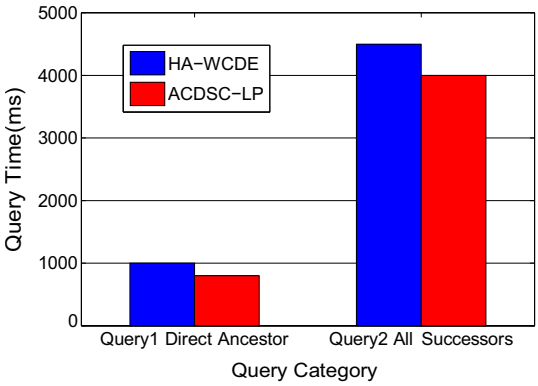

Fig. 7 Query time comparison

case, getting each node need to query ArtifactDB to find the integer number of node using the file name of node, and then query DictionaryDB to find the string of node using the integer number of node. Decompression can incur time overhead during the querying database. Along with the scaling-up number of recursive querying ancestor and successor, the number of decompression displays a upwardtendency, which adds more time overhead. Second, ACDSC-LP algorithm further reduces the size of each provenance record by only collecting basic provenance information, making the size of the records to be read much smaller than in the HA-WCDE case. This incurs a more efficient query. As we have stated above, we can see that the query performance of ACDSC-LP outperforming HA-WCDE.

6.4 Construction time of data supply chain

Based on provenance sets, we construct data supply chain using ACDSC-LP algorithm and PDRM algorithm, respectively. In our ACDSC-LP algorithm, by merging and sorting provenance records across different data platforms, DSCEM get the data supply chain. While PDRM algorithm uses provenance reconstruct the ancestor node. During rebuilding, the algorithm displays data supply chain by replaying the generation process. We set different data size (100 K, 1 MB, 10 MB, 100 MB, 500 MB). The construction time is composed of the time costing on provenance query and construction execution.

From the results reported in Fig. 8, the construction time of ACDSC-LP algorithm and PDRM algorithm increase with increasing of data size. ACDSC-LP algorithm significantly outperforms PDRM algorithm for construction time of data supply chain with different data size. The reason for the improvement is two-fold. First, PDRM algorithm may affects the normal node during data reconstruction. Typically, a rebuild process generates multiple nodes at a time, thus the generated nodes will automatically overlie the existing nodes. Obviously, some of these nodes exist with a higher version before rebuild will be covered. Hence, recovering the affected nodes to the normal state and executing the process that generates nodes and input parameters needed during the process execution make an obvious construction performance drop. Second, efficient

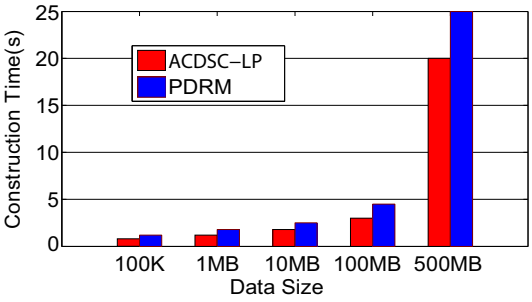

Fig. 8 Construction performance for ACDSC-LP and PDRM algorithms with provenance sets in different size

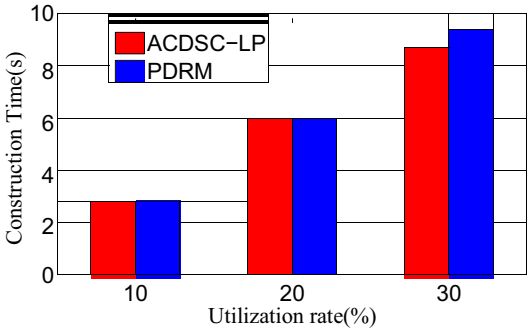

Fig. 9 Construction performance for ACDSC-LP and PDRM methods with different utilization rate of provenance sets

provenance query is a must for high performance construction. Above section illustrate that, for the ACDSC-LP algorithm case, querying provenance from database is more efficient and much faster. Based on the two above-mentioned methods, the total time in ACDSC-LP algorithm case is still smaller than in the PDRM algorithm case.

Figure 9 shows the construction time for ACDSC-LP and PDRM methods with different utilization rate of provenance sets. From the results reported in Fig. 9, the construction time of the two above-mentioned methods increases as the utilization rate improves. ACDSC-LP significantly outperforms PDRM for construction time of data supply chain with different utilization rate. The reason for the improvement is that PDRM may generate a series of nodes even though the number of real nodes needed to be rebuilt is only one or two. These excrescent nodes generated can raise the rebuild execution time and the provenance query time. In addition, ACDSC-LP compresses the provenance records in the PROV database and decreases the number of them, which reduces the time of query and construction execution to achieve better performance. We can see that provenance compression consumes less time for data supply chain construction. The reason is that the compression time is much smaller than the time costing on construction execution and provenance query, which illustrates validity of the proposed approach from the side.

7 Conclusion

In this paper, we constructed data supply chain effectively in distributed environments by proposing a scheme of constructing data supply chain based on layered PROV.

We propose a PROV data generation algorithm for multi-platform, which formally describes actions performed on data throughout its life cycle to achieve cross-platform provenance query. In addition, to manage provenance records with high efficiency, we introduce a tiered storage management of provenance based on summarization technology, which is based on processing provenance records elaborately, and compress the duplicate mid versions. A key innovation is to display a multi-level data supply chain using a layered query mechanism, which achieves high effective querying. The experimental results indicate that this algorithm achieves the best tradeoff on compression ratio, compression time, and construction time when compared with the existing algorithm.

In our future work, we intend to further reduce the collection overhead of SPADE system and improve the query performance for constructing data supply chain in distributed environment.

Acknowledgements This work is partly supported by the National Natural Science Foundation of China under Grant 61272520, 61370196, 61532012; the Research Fund for the Doctoral Program of Higher Education under Grant No.20110005110007.

References

1. Groth P (2013) Transparency and reliability in the data supply chain. Internet Comput IEEE 17(2):69–71
2. Zhou W, Fei Q, Narayan A et al (2011) Secure network provenance. The 23rd ACM Symposium on Operating Systems Principles (SOSP 2011), pp295–310, 23–26 October 2011
3. Xie Y, Feng D, Tan Z et al (2013) Design and evaluation of a provenance-based rebuild framework. IEEE Trans Magn 49(6):2805–2811
4. Stamatogiannakis M, Groth P, Bos H (2015) Looking inside the black-box: capturing data provenance using dynamic instrumentation. Provenance and Annotation of Data and Processes, vol 8628, pp 155–167
5. Ko RKL, Will M (2014) Progger: an efficient, Tamper-evident Kernel-space logger for cloud data provenance tracking. In: IEEE 7th International Conference on Cloud Computing (CLOUD). IEEE, New York, pp 881–889
6. Yu T, Ko RKL, Holmes G (2013) Security and data accountability in dis- tributed systems a: provenance survey. In: 2013 IEEE 10th International Conference On High Performance Computing and Communications 2013 IEEE International Conference On Embedded and Ubiquitous Computing (HPCC EUC). IEEE, New York, pp 1571–1578
7. Xie Y, Muniswamy-Reddy KK, Feng D et al (2013) Evaluation of a hybrid approach for efficient provenance storage[J]. ACM Trans Storage 9(4):1752–1756
8. Moreau L, Clifford B, Freire J et al (2010) The open provenance model core specification (vl.l). Future Gen Comput Syst 27(6):743–756
9. Moreau L, Missier P (2013) PROV-DM: the PROV data model. http://www.w3.org/TR/2013/REC-prov-dm-20130430/
10. Jones S , Strong C, Parker-Wood A, Holloway A, LongD D E (2011) Easing the burdens of HPC file management. PDSW '11 Proceedings of the sixth workshop on Parallel Data Storage, New York, NY, USA pp 25–30 November 2011
11. Mattoso M, Dias J, OcanaKary ACS et al (2015) Dynamic steering of HPC scientific workflows: A survey. Future Gen Comput Syst 46:100–113
12. Korolev V, Joshi A (2014) PROB: a tool for tracking provenance and reproducibility of big data experiments. The 20th IEEE International Symposium on High Performance Computer Architecture (HPCA2014), 02 March 2014
13. Imran A, Agrawal R, Walker J et al (2014) A layer based architecture for provenance in big data. In: 2014 IEEE International Conference on Big Data (big data). IEEE, New York, pp 29–31

14. Gehani A, Tariq D (2012) SPADE: support for provenance auditing in distributed environments. ACM/IFIP/USENIX 13th International Middleware Conference, pp 101–120, 3–7 December 2012
15. Zhao D, Shou C, Malik T et al (2013) Distributed data provenance for large-scale data-intensive computing. In: 2013 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, New York, pp 1–8
16. Suen CH, Ko RKL, Yu ST et al (2013) S2Logger: End-to-End Data Tracking Mechanism for Cloud Data Provenance. TrustCom2013:12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp 16–18 July 2013
17. Jacobson V, Braynard RL, Diebert T et al (2012) Custodian-based information sharing. IEEE Commun Mag 50(7):38–43
18. Zhang C, Li S (2016) Secure information sharing in internet-based supply chain management systems. J Comput Inf Syst 46(4):18–24
19. Freire J, Miles S, Missier P et al (2011) The open provenance model core specification (v1.1)[J]. Future Gen Comput Syst 27(6):743–756