A Hybrid Approach for Efficient Provenance Storage

Yulai Xie[†], Kiran-Kumar Muniswamy-Reddy[§], Dan Feng[†], Yan Li[‡], Darrell D. E. Long[‡], Zhipeng Tan[†], Lei Chen[†]

[†] *School of Computer, Huazhong University of Science and Technology*
Wuhan National Laboratory for Optoelectronics
[§] *Harvard University*
[‡] *University of California, Santa Cruz*
Email: ylxie@smail.hust.edu.cn, kiran@eecs.harvard.edu, dfeng@hust.edu.cn
{yanli, darrell}@cs.ucsc.edu, zhipengtan@163.com, chenlei5662@gmail.com

Abstract

Efficient provenance storage is an essential step towards the adoption of provenance. In this paper, we analyze the provenance collected from multiple workloads with a view towards efficient storage. Based on our analysis, we characterize the properties of provenance with respect to long term storage. We then propose a hybrid scheme that takes advantage of the graph structure of provenance data and the inherent duplication in provenance data. Our evaluation indicates that our hybrid scheme, a combination of web graph compression (adapted for provenance) and dictionary encoding, provides the best tradeoff in terms of compression ratio, compression time and query performance when compared to other compression schemes.

Categories and Subject Descriptors

E.4 [**Coding and Information Theory**]: Data compaction and compression; H.3.2 [**Information Storage and Retrieval**]: Information Storage-*File organization*

General Terms

Algorithms, Experimentation, Performance

Keywords

provenance graphs, storage, compression

1. INTRODUCTION

Provenance is the metadata that represents the history or lineage of a data object. Provenance has applications in various areas in the real world, such as experimental documentation, debugging, security, and search. The provenance community has built a number of systems [10, 12, 8] to collect provenance. While these systems are a great step towards making provenance available to users, they neglect a crucial aspect that makes these systems practical: efficient provenance storage. Unoptimized provenance storage can take up a substantial amount of space. For instance, the base data in the PReServ [6] provenance store was 100 KB, but the provenance exceeded 1 MB. In MiMI [7], an online database for storing protein information, the size of provenance (6 GB) also far exceeded the size of the original data (270 MB). Similar results are also observed in other systems [8, 11].

We cannot however, directly apply existing compression techniques to provenance, because the structure and access patterns of provenance are vastly different from regular metadata:

- Provenance needs to be queriable.
- Provenance is a graph that connects objects. Its structural characteristics are similar to web graphs.
- Lossy compression techniques are not applicable, because losing an edge will mean a disconnect in the provenance graph.
- Provenance can also have a great deal of duplication, implying that de-duplication can help us store provenance efficiently.

Accordingly, we developed a hybrid method for compressing provenance graphs by combining web graph based compression and dictionary based compression algorithms. The web graph compression algorithm allows us to compress provenance while still satisfying the characteristics we observed. Dictionary encoding, because of its flexible processing granularity, allows us to eliminate any repeated separate strings or sub-strings in the provenance graphs. Then, we compared the performance of our hybrid approach with a compression scheme designed explicitly for provenance compression: the Factorization And Inheritance (FAI) method proposed by Chapman et al. [4]. Our results indicate that the hybrid approach used by us significantly outperforms FAI along all axes.

2. CHARACTERISTICS OF PROVENANCE GRAPHS

2.1 Composition of provenance

A provenance graph is composed of two distinct parts: identity information on provenance nodes and ancestor information on provenance edge. We investigated their distribution in a large variety of provenance traces as shown in Table 1.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'12, October 29–November 2, 2012, Maui, HI, USA.
Copyright 2012 ACM 978-1-4503-1156-4/12/10 ...$15.00.

Table 1: Basic descriptions of a large variety of provenance traces. The table also summarizes the provenance systems that collect the traces, the percentage of size of identity and ancestor information in various provenance traces and where these traces can be found.

Provenance Trace	Description	Provenance System	Size of Identity %	Size of Ancestor %	Source
NAM-WRF	Weather and ocean modeling	Karma	68.12%	31.88%	[5]
NCFS	Weather and ocean modeling	Karma	60.58%	39.42%	[5]
SCOOP	Weather and ocean modeling	Karma	71.15%	28.85%	[5]
Gene2life	Bioinformatics and biomedical	Karma	70.43%	29.57%	[5]
Motif	Bioinformatics and biomedical	Karma	54.08%	45.92%	[5]
Animation	Computer animation rendering	Karma	59.21%	40.79%	[5]
J062941	Managing large-scale, complex scientific data and metadata collections	Tupelo [13]	33.49%	66.51%	[3]
pc3opm	A set of assertions made by the services involved in a process	PASOA [10]	33.21%	66.79%	[3]
OPMGraph-complete	A sequence of steps with exit conditions	Taverna [12]	33.92%	66.08%	[3]

It can be observed that there is no particular pattern in the distribution. In some traces, the provenance is dominated by the identity information and in others it is dominated by ancestry information. The identity information typically dominates in traces from Karma system. The reason is that those Karma traces are collected based on the Open Provenance Model (i.e., OPM [9]) and record a sizeable amount of property-values (though this is optional in OPM model) to describe the elements (e.g., workflowID and serviceID) in the identity information. Ancestry information takes up a larger percentage in the traces from Tupelo [13], PASOA [10] and Taverna [12]. These traces do not record the optional property-value pairs in the identity information, but still record the ancestor information, such as role, time and account, on the edges.

2.2 Duplication in provenance graphs

Duplication is widespread among annotation information in both of the identity information (e.g., account, type and workflowID) and ancestry information (e.g., account and Time) in some OPM traces. For example, the account, which is identical for every node and edge in an individual workflow, is recorded for every node and edge in OPM traces. The experimental results on those Karma traces in Section 4 show that the percentage of duplicates can be from 34.86% to 37.50% in the identity information, and 38.83% to 40.26% in the ancestor information of six workflow graphs in the Karma traces.

2.3 Similarity to web graphs

A web graph has a node for each URL and an edge for each hyperlink from one web page pointing to another. Existing web graph compression algorithms [1] typically exploit the following two key properties to significantly compress web graphs:

- **Locality:** Many links are within a URL domain, and therefore are not likely to point to pages far away.
- **Similarity:** Adjacent web pages have a high probability to have a common set of neighbors.

Provenance graphs also have similar structure properties as web graphs. Table 2 shows a series of provenance nodes from the NAM-WRF provenance trace presented as an adjacency list that represents the provenance graph. We assign an ID to each node according to their order in the provenance trace. Provenance nodes in this adjacency list have a common ancestor node 15. This is because many processes, represented by nodes like 15, trigger a new process (e.g., node 1) and generate a file (e.g., node 3). Further, some configuration files are also repeatedly used as input by many processes, therefore they appear as the common ancestors of many processes. These nodes also exhibit locality. For example, the ancestors of provenance node 1 are only between 2 and 16, and the ancestors of node 4 are only between 5 and 17. The reason is that we assign ID to these nodes based on the orders of their appearance in the trace, so one node has a high probability to be located not far from its ancestors.

Table 2: A slice of the adjacency list for the NAM-WRF provenance trace.

Node string	Node ID	Ancestors
Process_25367	1	2, 3, 15, 16
File_140	3	15
Process_25412	4	5, 6, 15, 17
File_230	6	15
......	......	......

Table 3: Statistics of provenance nodes on locality and similarity

Trace	No. of nodes in total	No. of local nodes (%)	No. of similar nodes (%)
NAM-WRF	65000	60000 (92.31%)	40000 (61.54%)
NCFS	4140	2530 (61.11%)	2990 (72.22%)
SCOOP	77000	70000 (90.91%)	35000 (45.45%)

Table 3 shows the number of provenance nodes that have these two properties in some provenance traces. We specify a node has the locality property (i.e., "local node") if the difference between its biggest and smallest ancestor is no larger than 15. A node has the similarity property (i.e.,

Table 4: Overview of various workflow characteristic in Karma trace

Workflow Name	Scientific domain	Failure rate	Number of Manipulations in a workflow	Number of Provenance Graphs	Total Graphs Size
NAM-WRF	Weather and ocean modeling	39%	6	5990	160MB
NCFS	Weather and ocean modeling	47%	7	298	10.3MB
SCOOP	Weather and ocean modeling	38%	6	7987	208MB
Gene2life	Bioinformatics and biomedical	46%	8	7990	257MB
Motif	Bioinformatics and biomedical	75%	138	498	294MB
Animation	Computer animation rendering	72%	22	430	45.6MB

"similar node") if this node shares at least one common ancestor with the node in preceding 10 nodes. One can see that all these traces have a large percentage of similar nodes and local nodes. This indicates that these provenance graphs also exhibit locality and similarity, like web graphs.

2.4 Provenance should be queriable

For provenance to be useful, it needs to be queriable, even when it is compressed. So a compression scheme that makes queries difficult or has a high query overhead is not preferred.

3. COMPRESSION ALGORITHMS

3.1 Web Compression Algorithms

There are two critical ideas incorporated in the current web compression algorithms [1]. First, *Exploitation of Similarity*, i.e., expressing the ancestor list of one node in terms of another node with similar ancestors, thus efficiently avoiding encoding the duplicate data. Second, *Exploitation of Locality*, i.e., rather than storing the ancestors of a node, storing the gaps between them, which typically requires fewer bits to be encoded.

We express a provenance graph as a set of provenance nodes which have a series of ancestors (See Table 2). Each provenance node is assigned an ID, in order of appearance, during provenance generation. Let Out(x) denote the ancestor list of node x and W indicate the window parameter. We detail the web compression algorithm as follows:

1. Reference compression: check if there exists a similar ancestor list in the preceding W ancestor lists. If y is such a reference node, $x - y$ is called reference number and Out(y) (ancestor list of y) is called reference list. We encode Out(x) into three parts: the reference number $x - y$, a sequence of bits that specify which ancestor in Out(y) also appears in Out(x), and the rest of the ancestors in Out(x).

2. Encode gaps: Let the ancestors of x yet to be encoded after the above step be x_1, x_2, x_3, ..., x_k. If $x_1 \leq x_2 \leq ... \leq x_k$, then we encode gaps $x_1 - x$, $x_2 - x_1$, ..., $x_k - x_{k-1}$.

Note that if we want to query the ancestor list (Out(x)) of node x, we have to first decode its reference list Out(y), and then we have to decode the reference list of Out(y), and so on. This would form a reference list chain, with a long chain obviously resulting in bad query performance. In our implementation, we confine the length of this chain to a maximum level of 5.

3.2 Dictionary Encoding

Dictionary encoding scans the entire database or text files to find the frequently occurring strings, and then replace them with integer codes.

For provenance, we look for frequently occurring strings in the provenance graph data, then we use integer codes to encode them and store this mapping relationship into a dictionary database. The granularity of the repeated strings is very flexible. It can be a whole string, the prefix of it or an arbitrary substring in a big string. For example, the edge in an OPM graph is usually annotated with time which indicates when this process (or dependency relationship) happens. We can use dictionary encoding to encode the common prefix that consists of year, month and day in the time information.

3.3 Combination of Web Compression Algorithms and Dictionary Encoding

As we have stated above, web compression algorithms can compress the ancestor information very effectively and dictionary encoding can eliminate the duplicates existing in the provenance graph data. Their combination provides a practical and efficient method to compress a provenance graph. Additionally, since web compression and dictionary encoding are both light-weight compression schemes, this hybrid method (i.e., web+dictionary) retains a good query performance on provenance datasets.

4. EVALUATION

4.1 Experimental Setup

The experiments were run on a machine with the Windows 7 (32-bit) operating system, Pentium(R) Dual-Core E6500 2.93 GHz*2 CPU, 2 GB memory and 500 GB hard drive.

The provenance traces we used were extracted from a 10GB noisy provenance database [5] generated by using the Karma [2] system. These traces consist of provenance generated from six kinds of workflows (See Table 4). The workflows are from different scientific domains and accordingly have different characteristics.

We compressed these traces using FAI and our hybrid method respectively. In both cases, we store the compressed provenance records in a MSSQL database (Microsoft SQL Server 8.0). Table 5 shows the schema that we used to store provenance records for the two techniques. In our hybrid method, for ancestor information, we assign each node string a unique identifier to make the web compression easier. Since a node can have a series of ancestors, we encode them into one "ancestors coding" using web compression algorithm and store each record in a concise format (i.e.,

Table 5: Database schema for web+dictionary and FAI techniques.

Compression techniques	provenance composition	databases	provenance records
web+dictionary	identity	ArtifactDB	(node string, account, size, fileURL)
		ProcessDB	(node string, account, workflowID, serviceID, timestep, workflowNodeID)
		DictionaryDB	(id, duplicate)
	ancestor	AncestorDB	(node ID, ancestors coding)
		NodeDB	(graphID, node ID, node string)
		TimeDB	(node string, times)
		RoleDB	(node string, Roles)
		AccountDB	(node string, accounts)
FAI	identity	ArtifactDB	(node string, account, size, fileURL)
		ProcessDB	(node string, account, workflowID, serviceID, timestep, workflowNodeID)
	ancestor	AncestorDB	(node string/node id, ancestor, role, account, time)
		DictionaryDB	(node id, duplicate)

Table 6: FAI techniques: resulting size (% of original)

FAI techniques	Nam-wrf	Ncfs	Animation	Gene2life	motif	scoop	total
Basic Factorization	100%	100%	100%	100%	100%	100%	100%
Node Factorization	100%	100%	100%	100%	100%	100%	100%
Argument Factorization	37.9%	41.7%	35.9%	45.0%	36.5%	42.8%	39.5%
Structural Inheritance	100%	100%	100%	100%	100%	100%	100%
Predicate Inheritance	78.75%	83.69%	81.80%	78.60%	83.33%	78.85%	80.31%
Argument Factorization and Predicate Inheritance	37.9%	41.7%	35.9%	45.0%	36.5%	42.8%	39.5%

(Node identifier, ancestors coding)) in AncestorDB. While FAI is composed of a series of factorization and inheritance methods (See Table 6), argument factorization achieves the best compression ratio; it finds the duplicate node strings in AncestorDB and encodes them using integer codes, and then stores them into DictionaryDB. We use FAI to represent argument factorization in the rest of the paper.

4.2 Compression Performance

Figure 1 shows the compression size and time for various workflow traces using FAI and the web+dictionary methods respectively. Web+dictionary outperforms FAI in both cases. The reason for the improvement of compression size is that FAI can only eliminate the duplicate node strings in the ancestor information, while web+dictionary seeks the locality and similarity between the ancestors of different nodes, and encodes all the ancestors that belong to a node to a 0/1 sequence. In addition, it reduces the duplicate strings in the annotation information in both ancestor and identity information. This exploits the redundancy in provenance graph to the maximum extent possible.

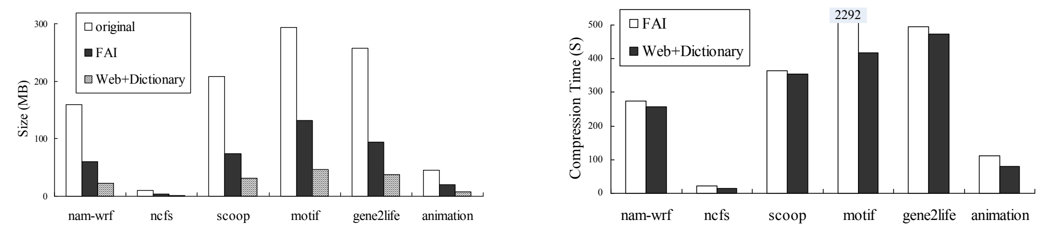
The reason for the improvement on compression time is two-fold. First, web+dictionary eliminates the duplicate strings in the identity information, making the size of the provenance records loaded to ArtifactDB and ProcessDB much smaller. In turn, this reduces the time needed to store the provenance records. Second, FAI employs a hash table to store the duplicate node strings and the frequency with which they appear in the edge (or ancestor) information. For provenance graphs that have a large number of nodes, such as motif, the time to query the hash table increases as the number of nodes increases. This indicates that FAI is not suitable for compressing large provenance sets.

4.3 Query Performance

To compare the query performance of compression methods, we ran a series of queries on the NAM-WRF trace as shown in Figure 2.

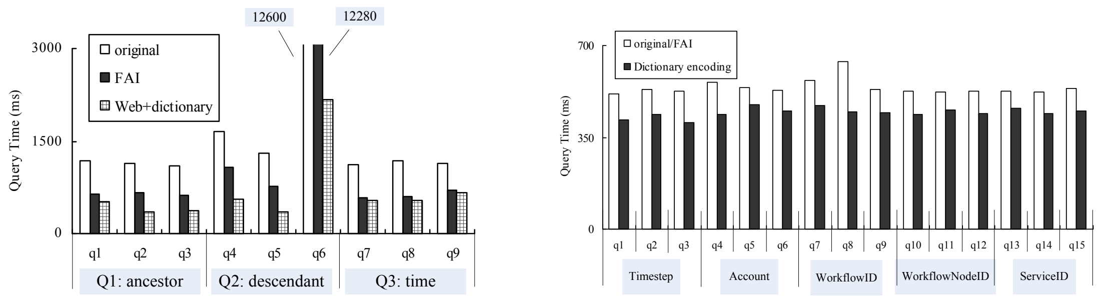
Q.1 looks up the ancestor of a specific object. The query process in Q.2 is similar to Q.1. However, the query has to repeat the step in Q.1 recursively until all the descendants have been located. In both cases, web compression performs better than FAI. The reason for the improvement is twofold. First, web compression significantly reduces the number of provenance record in AncestorDB. The query on the ancestor of a node will return a series of records in FAI, but only one record under web compression. Second, web compression algorithm further reduces the size of each record by exploiting the similarity and locality in the ancestors, making the size of the records to be read much smaller than in the FAI case. Though web decompression can incur time overhead during the ancestor queries, we have reduced this impact by confining the length of the provenance chain to 5 to avoid a limitless decompression.

In Q.3, the web+dictionary encoding also outperforms FAI. The improvement is because, for the web+dictionary encoding case, the TimeDB stores all the time information of a node string (these time information are on the edges that starts from this node string) in only one database record, while FAI uses one record for storing the time information on each edge in AncestorDB. So the number of the entries that needs to be queried in TimeDB in web+dictionary encoding is much fewer than in AncestorDB in FAI. On the other hand, the time information in TimeDB has been encoded using dictionary encoding, so the size of the record is much smaller than in the FAI case. Though querying the DictionaryDB incurs overhead, the total time in web+dictionary



(a) Compression size (b) Compression time

Figure 1: Compression performance for various workflow traces.



(a) Q.1-Q.3 (b) Q.4

Figure 2: Query performance. Q.1 looks up the ancestor of a specific object. Q.2 looks up all the descendants of a specific object. Q.3 looks up the *time* information on the edges. Q.4 looks up the element in identity information of ProcessDB. For each query, we present three sample results.

case is still smaller than in the FAI case. Similarly, in Q.4, the elements (such as *workflowID*) are encoded and stored using small integer numbers in ProcessDB in dictionary encoding. Hence the query time in ProcessDB is much smaller than in the uncompressed case and FAI. Despite the cost of querying the DictionaryDB for the final strings, the total query performance with dictionary encoding is better than the uncompressed and FAI case.

5. CONCLUSIONS

As provenance accumulates over time, it can occupy a significant amount of storage. Today, users have two non-options: archive it in a non-queriable format or discard the provenance. In this paper, we have addressed this crucial issue and have provided users with a practical solution for storing provenance efficiently. Our compression method (web+dictionary encoding) performs significantly better than the FAI algorithm in both compression and query performance.

Acknowledgements

This work was supported in part by the National Basic Research 973 Program of China under Grant No. 2011CB302301, 863 project 2009AA01A402, NSFC No. 61173043, 61025008, 60933002, 60873028, Changjiang innovative group of Education of China No. IRT0725. This work was also supported in part by the National Science Foundation under award IIP-0934401 and Department of Energy under Award Number DE-FC02-10ER26017/DE-SC0005417. We also thank the sponsors of the SSRC and CRIS, including the National Science Foundation, Los Almos National Laboratory, LSI, IBM Research, NetApp, Samsung Information Systems America, Seagate Technology, Northrop Grumman, Symantec, Hitachi, CITRIS, the Department of Energy Office of Science, the NASA Ames Research Center and Xyratex.

6. REFERENCES

[1] P. Boldi and S. Vigna. The webgraph framework I: Compression techniques. In *Proc. WWW'04*, 2004.

[2] B. Cao, B. Plale, G. Subramanian, E. Robertson, and Y. Simmhan. Provenance information model of karma version 3. In *Proc. SWF'09*, 2009.

[3] The third provenance challenge. http://twiki.ipaw.info/bin/view/Challenge/ParticipatingTeams3.

[4] A. P. Chapman, H. V. Jagadish, and P. Ramanan. Efficient provenance storage. In *Proc. SIGMOD*, 2008.

[5] Y.-W. Cheah, B. Plale, J. Kendall-Morwick, D. Leake, and L. Ramakrishnan. A noisy 10GB provenance database. In *Proc. of the 2nd International Workshop on Traceability and Compliance of Semi-Structured Processes, in conjunction with the 9th International Conference on Business Process Management*, 2011.

[6] P. Groth, S. Miles, W. Fang, S. C. Wong, K. Zauner, and L. Moreau. Recording and using provenance in a protein compressibility experiment. In *Proc. HPDC'05*, 2005.

[7] M. Jayapandian, A. P. Chapman, V. G. Tarcea, C. Yu, A. Elkiss, A. Ianni, B. Liu, A. Nandi, C. Santos, P. Andrews, B. Athey, D. States, and H. V. jagadish. Michigan molecular interactions (MiMI): Putting the jigsaw puzzle together. *Nucleic Acids Research*, 2007.

[8] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer. Provenance-aware storage systems. In *Proc. USENIX'06*, 2006.

[9] Open provenance model, 2008. http://twiki.ipaw.info/bin/view/OPM.

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

[11] Y. L. Simmhan, B. Plale, and D. Gannon. A framework for collecting provenance in data-centric scientific workflows. In *Proc. ICWS'06*, 2006.

[12] Taverna workflow management system. http://www.taverna.org.uk/.

[13] Tupelo semantic content repository. http://leovip217.ncsa.uiuc.edu/.